Proyecto final de carrera

Ingeniería Industrial

Universitat Politècnica de Catalunya

ESTUDIO Y DISEÑO DE UNA APLICACIÓN DE PLANIFICACIÓN, SUPERVISIÓN Y CONTROL DE FLOTAS DE ROBOTS MÓVILES

Memória

Autor

Eric MARTÍNEZ LARA

Tutor

Juan Carlos HERNÁNDEZ PALACÍN

Convocatoria

Junio de 2014

Agradecimientos

Debo agradecer a toda mi familia y amigos, que siempre han estado ahí apoyándome en todo momento. Sin su apoyo este proyecto no habría conseguido finalizarlo.

Pero mi principal agradecimiento, es para mí tutor, Juan Carlos. Que incluso estando enfermo siempre ha estado ahí, dándome todo el apoyo que he necesitado. Sin lugar a dudas, no hubiese sido posible realizar este proyecto sin su inestimable ayuda.

Índice de Contenidos

I. INTRODUCCIÓN 16

1. Objeto del proyecto 18

2. Justificación y motivación personal 19

3. Alcance 20

4. Estructura del documento 21

II. MARCO TEÓRICO 23

5. Evolución de la robótica móvil 25

6. Control y guiado de robots móviles 29

6.1. Introducción 29

6.1.1. Concepto de misión, navegación y operación 29
6.1.2. Esquemas de navegación en robots móviles. 30
6.2. Navegación 34

6.3. Sistemas de comunicación 37

6.3.1. Pirámide OSI 37
6.3.2. Metodología MAD-SMART 42
6.3.3. Tecnología OPC 45

7. Planificación de caminos 52

7.1. Introducción a la planificación de caminos 52

7.2. Problemática en la planificación de camino 53

7.2.1. Múltiples objetos móviles 54
7.2.2. Restricciones cinemáticas 54
7.2.3. Incertidumbre y objetos móviles 55
7.3. Métodos de planificación de trayectorias 56

7.3.1. Grafos de visibilidad 56
7.3.2. Diagramas de Voronoi 56
7.3.3. Roadmap Probabilístico (PRM) 58

7.3.4. Modelo del espacio libre 58
7.4. Algoritmos de planificación de caminos 60
7.4.1. SLAM (Simultaneous Localization and Mapping) 60
7.4.2. Algoritmo buscador de caminos de Floyd-Warshall 61
7.4.3. Campos potenciales 62
7.4.4. Algoritmo Dijkstra 64

8. Supervisión de procesos 70

8.1. Introducción a la supervisión de procesos 70
8.2. Los Sistemas SCADA 70
8.2.1. Introducción a los sistemas SCADA 70
8.2.2. Prestaciones, requisitos y módulos de un SCADA 72
8.2.3. Arquitecturas SCADA 73

III. MARCO APLICADO 77

9. Solución adoptada 79

9.1. Solución real/simulación 79
9.2. Arquitectura de software propuesta 83

10. Definición del problema a resolver 87

10.1. Descripción del layout 87
10.2. Simplificaciones impuestas 89

11. Explicación detallada de las aplicaciones que forman la solución propuesta. 92

11.1. Aplicación de Visual Basic .net 92
11.1.1. Algoritmo Dijkstra 92
11.1.2. Funcionamiento de la aplicación 96
11.1.3. Restricciones aplicadas al sistema 111
11.2. Base de datos SQL Server 114
11.2.1. Estructura de tablas 114
11.2.2. Creación de tablas dentro de una base de datos SQL 117
11.2.3. Procesos almacenados 120
11.3. Archivo de servicio OPC 123
11.3.1. Funcionamiento de KEPserverEX 5.0 123
11.3.2. Drivers estudiados 127
11.4. La aplicación SCADA 134

11.4.1. Generación de la pantalla SCADA 134
11.4.2. Funcionamiento de la aplicación 137

12. Conexión entre clientes y servidores 140

12.1. VB.net / SQL Server 140
12.2. SQL Server / KEPServerEX 5.0 143
12.3. VB.net / KEPServerEx 5.0 147
12.4. KEPServerEx 5.0 / SCADA de iFix 149

IV. RESULTADOS, CONCLUSIONES Y PROPUESTAS DE MEJORA 153

13. Ejemplo de utilización de la aplicación 155

14. Resultado y conclusiones 175

15. Propuestas de mejora 176

Referencias bibliográficas 178

Índice de figuras

Figura 1. Cohete v2 [1] 25
Figura 2. Imagen de Elsie sin cascaron [2] 25
Figura 3. A la izquierda, el robot Beast. A la derecha, Shakey [3], [4] 26
Figura 4. Robot RBX5 26
Figura 5. Prototipos de robot Honda. De izquierda a derecha, E0, p1 y ASIMO v2 [6] 27
Figura 6. A la izquierda, Sojourner. A la derecha, el rover Opportunity [7],[8] 28
Figura 7. AIBO de Sony 28
Figura 8. A la izquierda, Roomba. A la derecha, Automower [10], [11] 28
Figura 9. Esquema de la arquitectura de un robot móvil para realizar una misión 29
Figura 10. Estructura de control de navegación para un robot móvil 31
Figura 11. Esquema de navegación implantado en Blanche de AT&T 32
Figura 12. Esquema de navegación reactiva 33
Figura 13. Mapeado de un entorno generado mediante SLAM [15] 34
Figura 14.En verde la trayectoria del robot, en rojo la observada por los sensores.[15] 35
Figura 15. La pila OSI [16] 37
Figura 16. Transmisión de datos a través del modelo OSI [17] 39
Figura 17. Funcionalidad de la capa de sesión [17] 40
Figura 18. Ejemplo de trama dentro del protocolo Ethernet [17] 41
Figura 19. Comunicación directa, indicrecta y manejo de roles [18] 42
Figura 20. Conversación de agentes en un entorno de recolección de objetos [18] 43
Figura 21. Arquitectura del proyecto SMART [18] 44
Figura 22: Tipo de periféricos con su diferente tipo de conexión [19] 45
Figura 23: Dispositivos y propietarios [19] 46
Figura 24: Concepto OPC [19] 46
Figura 25: Visión de conectividad OPC [19] 47
Figura 26. Arquitectura Cliente-Servidor [19] 47
Figura 27. Anatomía conceptual de un Servidor OPC [20] 49
Figura 28. Anatomía conceptual de un Servidor OPC [20] 51
Figura 29. Problema básico de planificación de movimientos [23] 53

Figura 30. Ejemplo de grafo de visibilidad [25] 56
Figura 31. Ejemplo de diagrama de Voronoi [25] 57
Figura 32. Diagrama de Voronoi, retracción del espacio libre [25] 57
Figura 33. Roadmap Probabilístico [25] 58
Figura 34 Construcción de un CRG [25] 59
Figura 35. Relación entre módulos SLAM 60
Figura 36. Efecto del campo potencial sobre el robot [26] 62
Figura 37. Representación de la componente atractiva del campo potencial [25] 63
Figura 38. Representación de la componente repulsiva del campo potencial [28] 63
Figura 39. Problemática de los mínimos locales [28] 64
Figura 40. Ejemplo de grafo 65
Figura 41.Primera iteración del algoritmo Dijkstra 65
Figura 42. Segunda iteración del algoritmo Dijkstra 66
Figura 43. Tercera iteración del algoritmo Dijkstra 67
Figura 44. Cuarta iteración del algoritmo Dijkstra 68
Figura 45. Quinta iteración del algoritmo Dijkstra 68
Figura 46. Solución del ejemplo Dijkstra 69
Figura 47. Arquitectura SCADA [32] 71
Figura 48. Roto-Scan 79
Figura 49. Método de operación del roto-scan 80
Figura 50. Aplicación creada para la realización de este proyecto 80
Figura 51. Formulario para la creación de nuevos layouts 82
Figura 52. Formulario para trabajar con el layout predeterminado 83
Figura 53. Figura de SCADA en su funcionamiento sin robots en marcha. 84
Figura 54. Flota de robots dentro del OPC Server 85
Figura 55. Distribución de algunas de las tablas creadas para este proyecto 85
Figura 56. Diagrama de flujo del programa global 86
Figura 57. Aproximación inicial 87
Figura 58. Medidas de la nave industrial 87
Figura 59. Primer planteamiento de disposición interna 88
Figura 60. Estructura defintiva en planta del "layout" 89
Figura 61. Layout definitivo con direcciones implementadas 90

Figura 62. Aplicación RoboCaminos 92
Figura 63. Realización de búsqueda de camino óptimo 93
Figura 64. Secuencia de nodos que realizará el robot 94
Figura 65. Acceso a la segunda aplicación del algoritmo Dijkstra 94
Figura 66. Segundo punto de aplicación del algoritmo Dijkstra 95
Figura 67. Botón para volver a la selección de modo 96
Figura 68. Ayuda para cada paso 97
Figura 69. Formulario de creación de un nuevo "layout" 98
Figura 70. Guía interactiva de pasos a seguir por el usuario 99
Figura 71. Primer bloque de pasos a seguir 100
Figura 72. Inicio de selección de nodos 100
Figura 73. Ayuda para cada paso modificada 100
Figura 74. Layout con los nodos seleccionados 101
Figura 75. Activación del segundo bloque de pasos a seguir 102
Figura 76. Pregunta sobre la existencia de camino entre nodos 102
Figura 77. Ayuda para cada paso modificada (2) 103
Figura 78. Retroceso del último rest point marcado 103
Figura 79. Marcar camino entre nodos 104
Figura 80. Paso al siguiente tramo 104
Figura 81. Grafo para el desarrollo del ejemplo Dijkstra 105
Figura 82. Pop-up de inicio del tercer bloque 105
Figura 83. Cambio al tercer bloque de la guía paso a paso 106
Figura 84. Ayuda para cada paso modificada (3) 106
Figura 85. Solución del algoritmo Dijkstra 107
Figura 86. Diagrama de flujo de la aplicación de creación de un layout 108
Figura 87. Bloque 'Pulsar "rest points" entre nodos (odometria) 108
Figura 88. Selección de "layout" propio 109
Figura 89. Formulario de utilización de "layout" propio 109
Figura 90. Diagrama de flujo de la aplicación de utilización de un layout predefinido 110
Figura 91. Mensaje de error, robots ocupados 111
Figura 92. Falta realizar la selección de nodos 112
Figura 93. Falta realizar el paso previo 112

Figura 94. Error de paso realizado 113
Figura 95. Error para realizar Dijkstra 113
Figura 96. Error al haber finalizado la selección de caminos 113
Figura 97. Estructura de tablas dentro de la base de datos SQL 114
Figura 98. Diagrama de flujo de la lógica de la programación SQL 116
Figura 99. Bloque 'Chequeo en tablas A y B para cada pareja de nodos' Ampliado 116
Figura 100. Creación de una base de datos SQL Server 117
Figura 101. Método sencillo para creación de tablas SQL 118
Figura 102. Ejemplo método sencillo para creación de tablas SQL 118
Figura 103. Método complejo de generación de tablas en SQL Server 119
Figura 104. Ejecutar para que la tabla se genere en el sistema 119
Figura 105. Estructura de procesos almacenados 120
Figura 106. Creación de un proceso almacenado 121
Figura 107. Consulta de creación de un proceso almacenado 121
Figura 108. Proceso almacenado volcar_datos 122
Figura 109. Estructura de dispositivos en KEPServerEx 123
Figura 110. Botón para la creación de nuevos canales 123
Figura 111. Nombre del nuevo canal 124
Figura 112. KEPServerEx selección de driver 124
Figura 113. Creación de canal finalizada 125
Figura 114. Generación de un dispositivo 126
Figura 115. Opción de tiempo de refresco de la tabla 126
Figura 116. Creación de un dispositivo dentro del canal 127
Figura 117. Selección del driver DDE Client 128
Figura 118. Propiedades de comunicación de un dispositivo DDE Client 129
Figura 119. Selección de driver Memory Based 131
Figura 120. Memory Based driver - Activar la persistencia de objetos 132
Figura 121. Generación de una nueva imagen en SCADA 134
Figura 122. Insertar una imagen en SCADA 135
Figura 123. Imagen introducida en SCADA 135
Figura 124. Distribución de objetos en el layout de SCADA 136
Figura 125. Distribución definitiva de objetos sobre el layout de SCADA 136

Figura 126. Envío de pedido 137
Figura 127. Un robot en funcionamiento 138
Figura 128. Dos robots funcionando simultáneamente 138
Figura 129. Funcionamiento del sistema con tres robots simultáneos 139
Figura 130. Añadir un nuevo objeto al proyecto 140
Figura 131. Selección del tipo de objeto DataSet 141
Figura 132. Inserción de un adaptador de tabla al objeto de prueba creado 141
Figura 133. Configuración de la base de datos 142
Figura 134. Estructura de tablas insertadas en VB.net desde SQL 143
Figura 135. Acceso a Origenes de datos (ODBC) 143
Figura 136. Creación de un nuevo origen de datos SQL Server 144
Figura 137. Proceso de creación de origen finalizado 144
Figura 138. Selección de la fuente de datos 145
Figura 139. Selección de tabla para cada dispositivo 146
Figura 140. Generación de tags de forma automática 146
Figura 141. Inserción de DA Junction en el formulario de VB 147
Figura 142. Acceso a la configuración de ClientAce 148
Figura 143. Configuración de ClientACE DA Junction 148
Figura 144. Animación de objetos en SCADA 149
Figura 145. Opciones de animación para los objetos de SCADA 150
Figura 146. Configuración de visibilidad de objetos 150
Figura 147. Selección de fuentes de datos 150
Figura 148.Servidor de datos Flota de robots dentro del configuración de animación .. 151
Figura 149. Introducción del primer pedido 155
Figura 150. Ejemplo de utilización de la aplicación(1) 156
Figura 151. Ejemplo de utilización de la aplicación(2) 156
Figura 152. Ejemplo de utilización de la aplicación(3) 157
Figura 153. Ejemplo de utilización de la aplicación(4) 157
Figura 154. Ejemplo de utilización de la aplicación(5) 158
Figura 155. Ejemplo de utilización de la aplicación(6) 158
Figura 156. Ejemplo de utilización de la aplicación(7) 159
Figura 157. Ejemplo de utilización de la aplicación(8) 159

Figura 158. Ejemplo de utilización de la aplicación(9) 160
Figura 159. Introducción del segundo pedido 160
Figura 160. Ejemplo de utilización de la aplicación(10) 161
Figura 161. Ejemplo de utilización de la aplicación(11) 161
Figura 162. Ejemplo de utilización de la aplicación(12) 162
Figura 163. Ejemplo de utilización de la aplicación(13) 162
Figura 164. Ejemplo de utilización de la aplicación(14) 163
Figura 165. Ejemplo de utilización de la aplicación(15) 163
Figura 166. Ejemplo de utilización de la aplicación(16) 164
Figura 167. Ejemplo de utilización de la aplicación(17) 164
Figura 168. Ejemplo de utilización de la aplicación(18) 165
Figura 169. Ejemplo de utilización de la aplicación(19) 165
Figura 170. Envío de un tercer pedido 166
Figura 171. Ejemplo de utilización de la aplicación(20) 166
Figura 172. Ejemplo de utilización de la aplicación(21) 167
Figura 173. Ejemplo de utilización de la aplicación(22) 167
Figura 174. Ejemplo de utilización de la aplicación(23) 168
Figura 175. Introducción del cuarto y último pedido 168
Figura 176. Ejemplo de utilización de la aplicación(24) 169
Figura 177. Ejemplo de utilización de la aplicación(25) 169
Figura 178. Ejemplo de utilización de la aplicación(26) 170
Figura 179. Ejemplo de utilización de la aplicación(27) 170
Figura 180. Ejemplo de utilización de la aplicación(28) 171
Figura 181. Ejemplo de utilización de la aplicación(29) 171
Figura 182. Ejemplo de utilización de la aplicación(30) 172
Figura 183. Ejemplo de utilización de la aplicación(31) 172
Figura 184. Ejemplo de utilización de la aplicación(32) 173
Figura 185. Ejemplo de utilización de la aplicación(33) 173
Figura 186. Ejemplo de utilización de la aplicación(34) 174
Figura 187. Ejemplo de utilización de la aplicación(35) 174

Parte I

INTRODUCCIÓN

1. Objeto del proyecto

El objeto de este proyecto reside en el estudio de la planificación, el control y la supervisión de una flota de robots móviles. En otras palabras, se requiere automatizar el proceso logístico dentro de un almacén de dimensiones determinadas, valiéndose de un número determinado de robots.

2. Justificación y motivación personal

Realizando el proyecto que se presenta, se pretende ampliar la base de conocimientos tecnológicos correspondiente a la parte más enfocada hacia la incorporación al mundo laboral.

La elección de este proyecto fue el resultado de la mezcla del interés por el aprendizaje de tecnologías y de sistemas completamente desconocidos para un ingeniero industrial estándar, y el interés que había sido generado por la asignatura Automatización Industrial (una de las asignaturas de la especialidad Eléctrica), en la cual, se da al alumno una pequeña introducción al mundo de la automática y de la automatización de procesos.

La motivación personal que ha ayudado al estudiante a llevar a cabo este proyecto ha sido el afán de superación de uno mismo, puesto que prácticamente la totalidad de los aspectos, en este proyecto tratados, son desconocidos para cualquier estudiante de Ingeniería Industrial.

3. Alcance

Los objetivos a cubrir con la realización de este proyecto son:

- Estudio de la solución que debería adoptarse en un supuesto caso real.
- Diseño de un software que simule la casuística propuesta previamente en el objeto, la automatización de la logística llevada a cabo dentro de una nave industrial.
    - Diseño y explicación detallada de una aplicación que simule para una distribución de planta determinada el funcionamiento automatizado del proceso logístico llevado a cabo dentro de la nave industrial. Incluyendo para dicha aplicación un visualizador del proceso mediante un sistema HMI SCADA.
    - Diseño y explicación detallada de una aplicación que permita adaptar la solución para cualquier distribución de entradas, salidas o zonas de almacenamiento interiores para una nave industrial de medidas determinadas. Excluyendo de esta solución la simulación con autómatas y el visualizador.

4. Estructura del documento

El documento se ha dividido en cinco secciones diferenciadas:

- **Introducción:**

  Es la sección actual, es allí donde se declara el objeto del proyecto, así como la justificación, el alcance y las la estructura que tendrá todo el documento.

- **Marco teórico:**

  En la segunda sección, se muestran todos los conceptos teóricos necesarios para la comprensión y realización de la siguiente sección del proyecto.

- **Marco aplicado:**

  En la tercera sección, se muestra la solución adoptada para la realización de este proyecto. Además de una explicación detallada del funcionamiento interno de las aplicaciones utilizadas, también incluye información de la creación del diseño y de la arquitectura de software.

- **Resultados, conclusiones y propuestas de mejora**

  En esta sección se realiza un ejemplo práctico paso a paso del funcionamiento (exteriormente) de la aplicación de simulación. Por otro lado se incluyen las conclusiones alcanzadas y las propuestas de mejora para próximos estudios en este campo.

- **Referencia bibliográfica**

  Contiene un largo listado con cada una de las fuentes consultadas para el aprendizaje y la solución de este proyecto.

Parte II

MARCO TEÓRICO

5. Evolución de la robótica móvil

Los primeros robots móviles aparecen a mediados del siglo XX durante la segunda guerra mundial. Estos emergieron debido al incremento de avances tecnológicos en los ámbitos de ciencia de la computación y en la cibernética. Mayoritariamente estos se trataban de distintos tipos de bombas, algún ejemplo de esto podrían ser las bombas inteligentes con un sistema de detección de rango que explotaban con la proximidad. Este tipo de bombas fueron las precursoras de los cohetes V1 y V2 que contaban con un piloto automático y con un sistema de detonación también automático. Estos últimos fueron los predecesores de los actuales misiles de crucero.

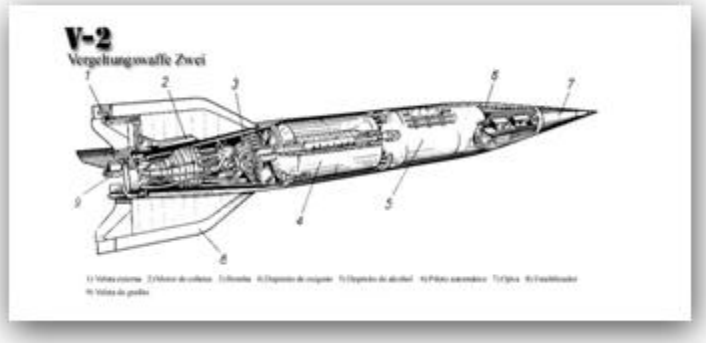

Figura 1. Cohete v2 [1]

Seguidamente, en el año 1948 William Grey Walter construyo Elmer y Elsie, dos robots autónomos llamados "Machina Speculatrix" porque estos robots estaban creados para explorar su entorno.

Figura 2. Imagen de Elsie sin cascaron [2]

Elsie y Elmer estaban equipados con unos sensores de luz, de manera que si estos encontraban un foco de luz se dirigían hacia éste, evadiendo en el camino cualquier tipo de obstáculos. Estos robots fueron la demostración de que a partir de un diseño sencillo podía surgir un comportamiento complejo.

Posteriormente, entre 1961 y 1963 la Universidad John Hopkins desarrolló "Beast", éste se ayudaba de un sonar para moverse y cuando su batería se agotaba, "Beast" iba en busca de un enchufe para auto-recargarse. Y a finales de los 60 "Shakey" fue desarrollado por el Instituto de Investigaciones de Stanford. Éste robot contaba con una cámara, un detector de rango, con diversos sensores de impacto y con conexión radio. "Shakey" era capaz de razonar sobre sus actos, es decir, éste únicamente necesitaba una orden genérica para desarrollar todos los pasos necesarios hasta cumplir su objetivo.

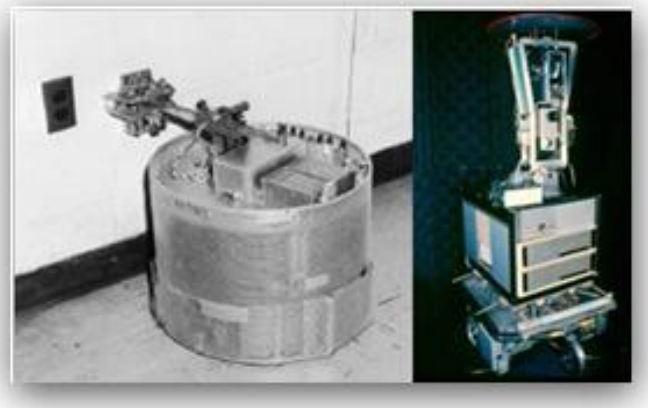

Figura 3. A la izquierda, el robot Beast. A la derecha, Shakey [3], [4]

A partir de este momento el interés por los robots móviles creció considerablemente, dando como resultado la comercialización de robots para uso doméstico. Estos robots sirvieron tanto como para entretenimiento como para fines educativos. Fue así como apareció la serie HERO, que a día de hoy todavía existe, y también modelos de robot como el RBX5 sacado al mercado por RC (Robot Corporation) el 1984.

Figura 4. Robot RBX5

El año 1986 la compañía nipona Honda creó el primero de una larga serie de prototipos de robots, el E0. Éste robot era un robot bípedo capaz de moverse en línea recta, precisaba aproximadamente de 5 segundos para dar cada paso. Al E0 le siguieron nuevas generaciones de prototipos que iban introduciendo mejoras. En 1993 el E6 era capaz de caminar de forma autónoma y a velocidad humana a la vez que sorteaba obstáculos sencillos. No obstante, todavía carecía de tronco y extremidades superiores. Elementos que fueron incorporados a la siguiente generación de prototipos P1,P2 y P3. Y fue para el año 2000 cuando Honda presentó la primera versión de ASIMO que era más ligero y flexible que los demás prototipos, contando con unos movimientos mejorados y con un aspecto más amigable.

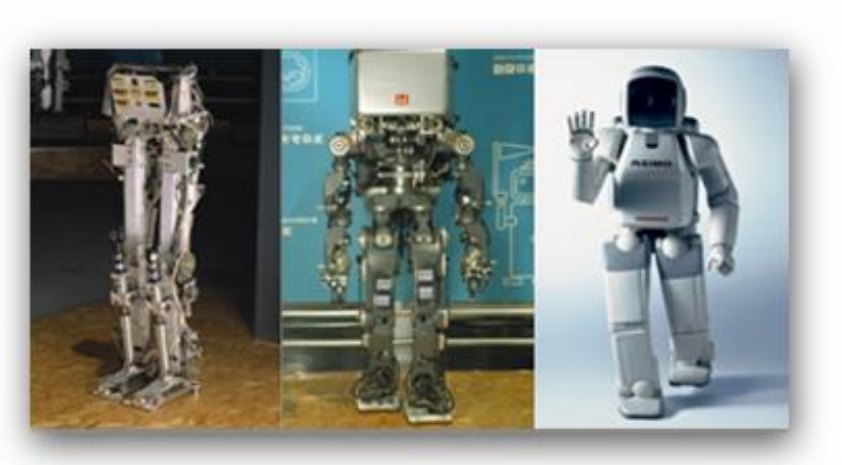

Figura 5. Prototipos de robot Honda. De izquierda a derecha, E0, p1 y ASIMO v2 [6]

Por otro lado, los robots móviles han sido también creados con fines prácticos. Tales como la exploración de otros planetas. Con ese fin en 1997fue creado el rover "Sojourner", que se convirtió en el primer robot en pisar la superficie de Marte. Estaba controlado mediante control remoto desde la Tierra, aunque incluía navegación autónoma utilizando un laser para detectar cualquier tipo de obstáculo. Y para el año 2004 se enviaron a Marte los rovers gemelos "Opportuniy" y "Spirit" englobados dentro de la misión de exploración Mars Exploration Rover. Son un modelo tecnológicamente más avanzado que el "Sojourner" puesto que dispone de un sistema de navegación más avanzado. A día de hoy estos dos rover todavía se encuentran en la superficie del planeta Rojo.

Figura 6. A la izquierda, Sojourner. A la derecha, el rover Opportunity [7],[8]

De cara al 1999, Sony lanzó al mercado AIBO, un perro robótico capaz de ver, caminar e interactuar con el entorno. Este incorporaba una gran cantidad de sensores que le hacían poder actuar en función a la experiencia que iba adquiriendo.

Figura 7. AIBO de Sony

I en estos últimos años se han empezado a fabricar robots para uso doméstico. A destacar, el robot limpiador Roomba de iRobot comercializado el añ y el robot cortador de cesped Automower de Husqvana.

Figura 8. A la izquierda, Roomba. A la derecha, Automower [10], [11]

6. Control y guiado de robots móviles

6.1. Introducción

6.1.1. Concepto de misión, navegación y operación

Se define navegación como el método a utilizar para guiar a un robot móvil dentro de un ambiente con obstáculos. Se puede hallar distintos tipos de esquemas, a pesar de que todos ellos centran sus esfuerzos en tratar de llevar al robot a su destino de forma óptima.

Así pues, un robot móvil se caracteriza por llevar a cabo un desplazamiento desde un punto a otro (navegación), mientras a su vez interactúa con el entorno sorteando cualquier clase de obstáculos (operación). Este hecho implica implícitamente el llevar a cabo una serie de objetivos exigidos por algún tipo de especificación. Entonces, dentro del ámbito de la robótica móvil se entiende como misión la realización combinada de unos determinados objetivos tanto de navegación como de operación.

Siguiendo las definiciones dadas en el párrafo anterior, se puede concluir que un robot móvil debe incluir una serie de elementos (sistema de sensores, control de movimiento y operación), y debe poseer una arquitectura capaz de coordinarlos de una forma óptima, para llevar a cabo eficientemente los objetivos de la misión. En la figura 9, que se muestra a continuación, puede apreciarse un esquema básico de los módulos principales que componen la arquitectura de un robot. Aunque cabe añadir que el diseño final de esta arquitectura dependerá finalmente de la aplicación que este vaya a desarrollar.

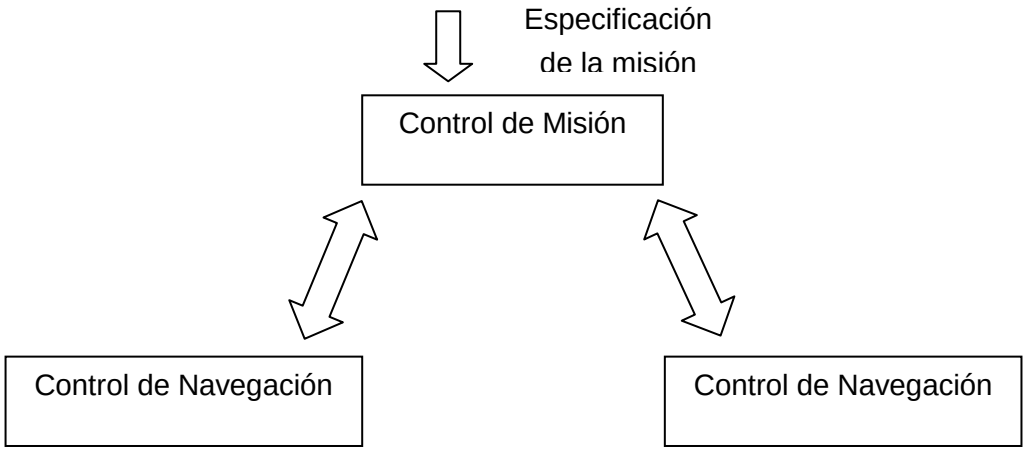

Figura 9. Esquema de la arquitectura de un robot móvil para realizar una misión

En el esquema anterior puede observarse que el módulo de control de misión, que es el encargado de estudiar el problema para resolverlo mediante la estrategia óptima, es el elemento empleado para coordinar tanto el control de navegación como el control de operación. Así pues, el módulo de control de misión se encargará de hacer llegar a los otros dos módulos cuales serán las acciones a realizar por cada uno.

6.1.2. Esquemas de navegación en robots móviles.

La navegación en robots móviles consiste básicamente en hacer que los robots se desplacen desde una localización inicial hasta una final. Por lo tanto se puede subdividir el problema de la navegación en cuatro diferentes etapas:

- **Percepción del mundo:** Consiste en el desarrollo de un mapa o de un modelo de entorno donde la navegación será efectuada, esto se consigue mediante el uso de diferentes sensores.

- **Planificación de la ruta:** Mediante las especificaciones de la tarea que se debe realizar, la utilización del mapa desarrollado con anterioridad y algún tipo de recurso estratégico, se calcula que sucesión ordenada de objetivos o submetas deben ser alcanzadas por el robot móvil.

- **Generación del camino:** En función a la ruta planificada con anterioridad, se discretiza la función desarrollada en la planificación de la ruta con el fin de generar el camino a seguir por el robot.

- **Seguimiento del camino:** Utilizando el camino generado en el paso previo se efectúa el desplazamiento del robot mediante un uso correcto del sistema de actuadores que éste incorpore.

Relacionando estas cuatro etapas se forma la estructura básica del módulo de control de navegación que se ha visto en la figura 9. Estas tareas previamente comentadas pueden realizarse por separado, aunque necesariamente en el orden que se ha estipulado. La interrelación entre estas tareas se muestra en la figura 10.

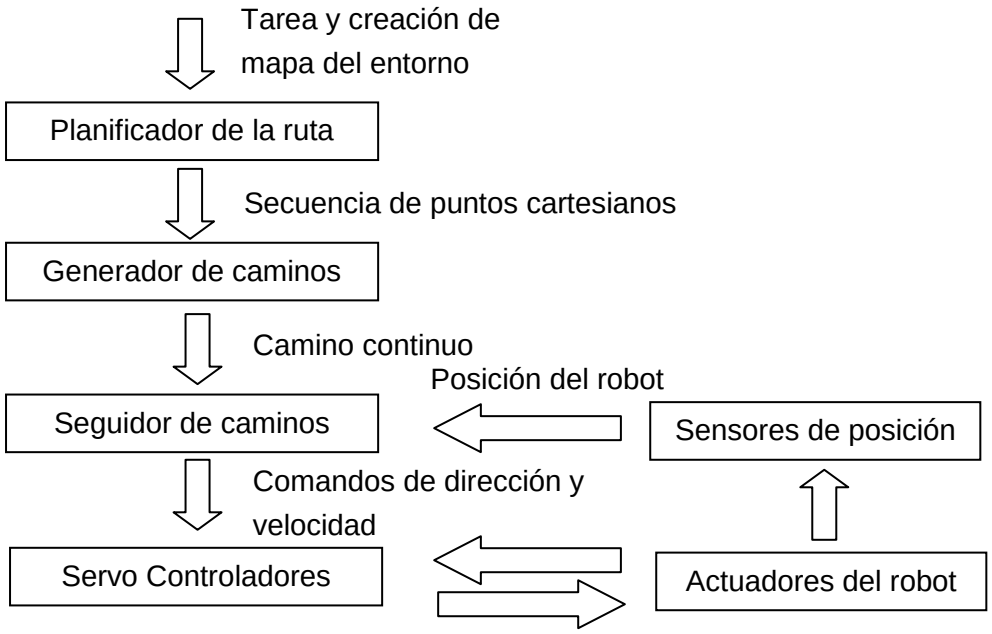

Figura 10. Estructura de control de navegación para un robot móvil

Para poder realizar esta tarea de forma efectiva, se necesita un conocimiento fiel del entorno de trabajo. De este modo, la figura anterior da por hecho que se posee un mapa del entorno que responde de forma fiel a la realidad. De este modo, se hace relativamente más sencilla la generación del camino que cumpla con todas las submetas propuestas al planificar la ruta, evadiendo al paso del robot móvil cualquier posible obstáculo que pueda hallar.

Debido al hecho que la figura 10 precisa de ese conocimiento exhaustivo del entorno, hace que el esquema presentado resulte no del todo eficaz, puesto que resulta relativamente sencillo que el robot posea un modelo del entorno con diversos defectos. Es por este motivo que existe la necesidad de introducir al esquema básico algún elemento nuevo que mitiguen los efectos de este defecto. Así fue como apareció el esquema mostrado en la figura 11, el cual es un sistema que se utiliza en aplicaciones donde no se posee toda la información del entorno de trabajo. Este sistema corresponde al implantado en el robot móvil Blanche en los laboratorios AT&T.

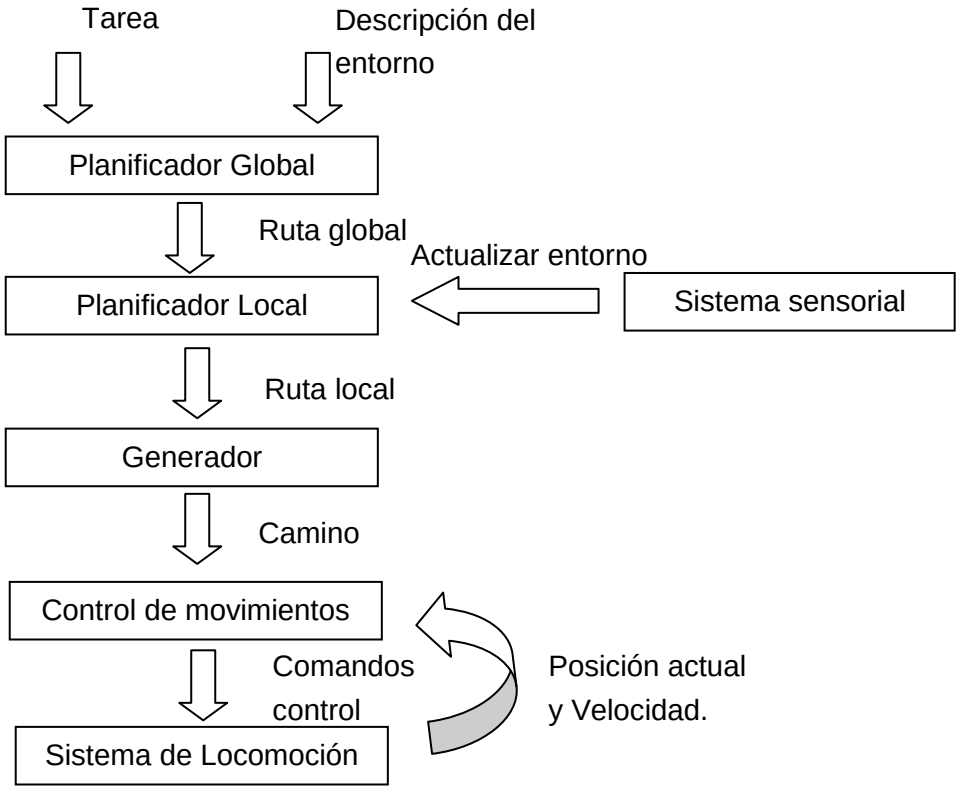

Figura 11. Esquema de navegación implantado en Blanche de AT&T

Lo novedoso de este sistema radica en el desdoblamiento de la planificación en dos diferentes subtareas: planificación global y local. La primera corresponde a la planificación que se efectuaba en el esquema básico. Éste construye una ruta con los conocimientos del entorno que posee a priori, definiendo un camino que en el momento inicial se supone libre de obstáculos. Mientras que la segunda, planificación local, consiste en un modelo que se realiza del entorno cercano al robot, utilizando los datos que se reciben mediante el sistema sensorial. De esta manera es posible replanificar una ruta local del robot y así evitar posibles colisiones.

Si se realiza una navegación en un entorno completamente conocido, resultaría redundante realizar la planificación local. Así pues, mientras el grado de incertidumbre del lugar va en aumento, la relevancia de realizar esta segunda planificación también incrementa.

La alternativa a este tipo de navegación estratégica reside en el uso masivo de sensores de coste bajo, ésta táctica hace que sea innecesario el uso de ningún tipo de planificación o seguimiento de caminos, ya que el robot es capaz de reaccionar de forma dinámica ante cualquier pequeño cambio del entorno. Este

modelo se basa en *subsumption arquitecture* (Brooks, 1986), esta arquitectura se basa en una separación vertical del problema de navegación. Es una arquitectura basada en módulos que están especializados para realizar tareas de forma individual. El esquema utilizado, denominado navegación reactiva, es el que se muestra en la figura

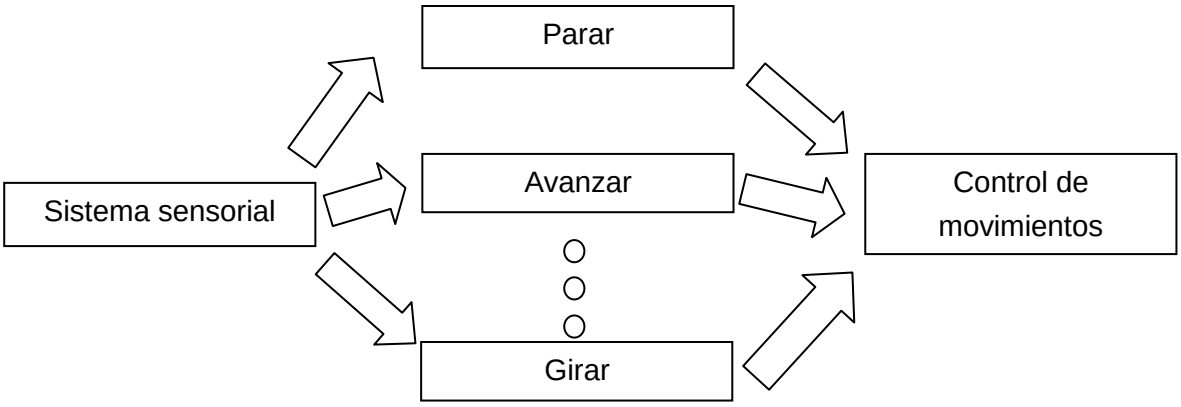

Figura 12. Esquema de navegación reactiva

En función de la información que se recibe a través del sistema sensorial, se activan tantos comportamientos simples como sean necesarios para realizar la siguiente acción, de este modo el comportamiento final es el resultado de mezclar todos los comportamientos simples activos.

Este tipo de navegación, se ha implantado en diversas aplicaciones, entre ellas cabe destacar el comportamiento de supervivencia, ya que el robot que dispone de este tipo de esquema se acaba convirtiendo en un robot errante vagando por libertad por un entorno dinámico. No obstante, raramente obedecen a algún pan preestablecido, cosa que resulta más que imprescindible si se trata de una misión real.

6.2. Navegación

El principal inconveniente para un robot para moverse de manera autónoma es debido a la determinación adecuada de su localización. Así pues, el problema de navegación consiste en buscar solución a dos preguntas clave.

(*¿Dónde estoy?*).

No es posible dar una localización precisa sin una idea adecuada del entorno que rodea al robot, un mapa que muestre toda clase de obstáculos que el autómata pueda encontrar en el camino. Así como toda clase de puntos fijos que le sirvan de referencia para poder localizarse adecuadamente.

(*¿Como es el mundo que nos rodea?)*

El principal problema reside en el hecho que los sensores odométricos son muy poco precisos, y que son la principal fuente de datos que utilizan los robots para conocer su posición. Esta situación obliga a utilizar otro tipo de sensores ópticos cuya precisión resulta mayor, estimando dichos datos con más certeza.

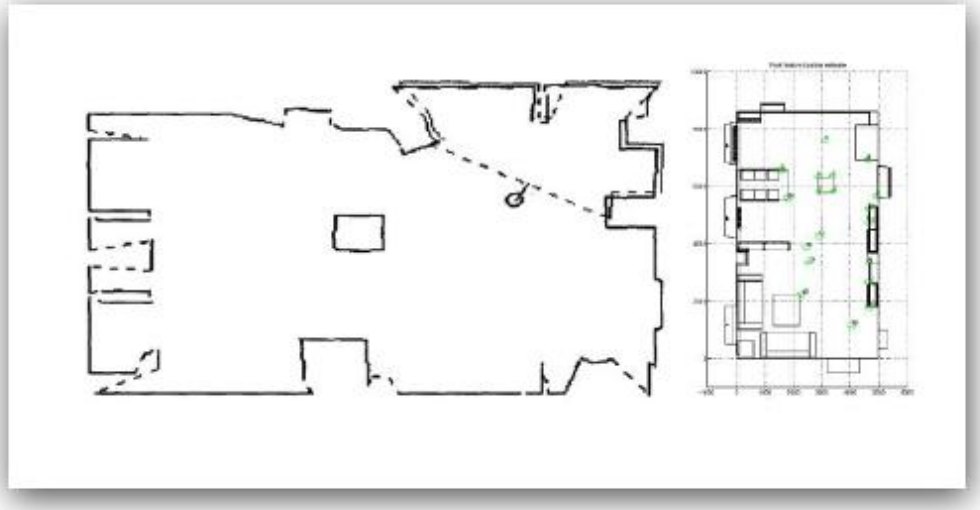

Figura 13. Mapeado de un entorno generado mediante SLAM [15]

Hoy en día, las opciones más utilizadas consisten en introducir en los robots previamente una especie de mapas "preconstruidos", o bien a medida que los robots van desplazándose van elaborando dichos mapas. Esta segunda es la alternativa más explotada durante estos últimos años y se le da el nombre de SLAM (*Simultaneous Localization and Mapping*).

Actualmente los principales esfuerzos en investigación en el campo de la robótica móvil se están centrando en desarrollar la solución al problema SLAM, puesto que está considerado como el punto clave para la plena autonomía de los autómatas.

La razón por la cual se están encontrando diversas dificultades, reside en la existencia de perturbaciones en las medidas aportadas por los sensores y en la limitación del rango que estas tienen. Realizando un estudio un poco más profundo de dicha problemática, los principales factores que dificultan este sistema son los siguientes:

- Las observaciones vienen dadas con respecto al sistema de referencia que posee el robot móvil, y la posición de este se encuentra afectada por la imprecisión adherente a la odometría (tal y como podemos comprobar en la figura 14. Así pues, los errores se maximizan, puesto que al error generado por la incertidumbre de la odometría se le añade la posición relativa del robot.

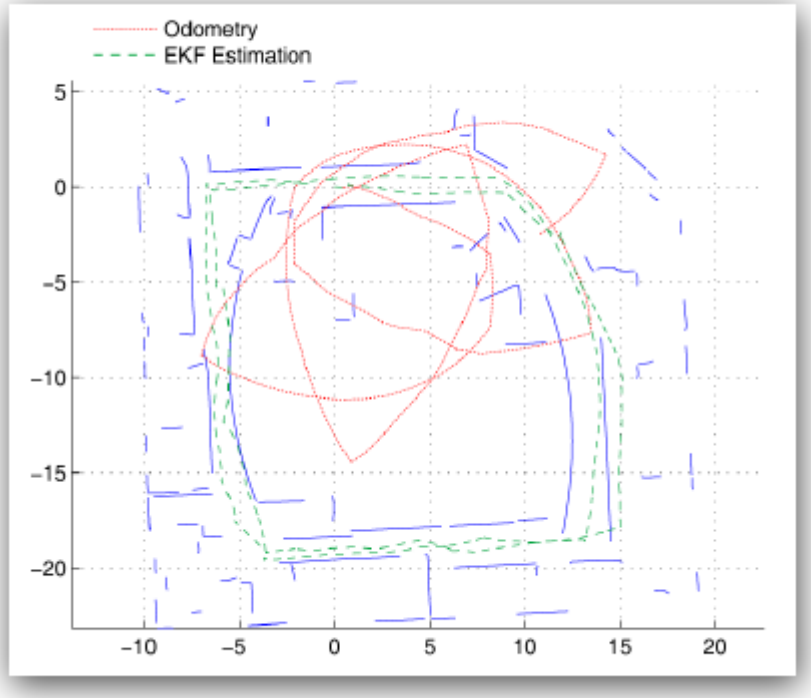

Figura 14.En verde la trayectoria del robot, en rojo la observada por los sensores.[15]

- En la mayoría de las ocasiones se deberían de generar unos mapas de tamaño excesivamente grande. Esto supondría un coste mayor computacional y por tanto un extra de imprecisión a la hora de realizar estimaciones odométricas según el robot se desplaza.

- Los entornos en los que se desplazan los robots, son usualmente dinámicos, mientras que las observaciones están realizadas a partir de puntos fijos en el mapa, lo cual simplifica mucho el problema. Una aproximación que funciona eficientemente consiste en ir borrando los objetos transitorios a medida que van desapareciendo del campo de visión del robot, como si de perturbaciones se tratase. No obstante, la ciencia de la visión artificial y la sensórica avanza rápidamente, lo que conllevará a que en poco tiempo se puedan crear estructuras estáticas dentro de un marco de referencia móvil.
- Los entornos de trabajo son tridimensionales, pero tener en cuenta este aspecto introduciría mucha complejidad al problema puesto que los sensores usualmente están configurados para trabajar en un plano horizontal. El paso a modelos en tres dimensiones es un objetivo a seguir en este campo.

6.3. Sistemas de comunicación

6.3.1. Pirámide OSI

A finales de la década de los setenta se empezó a desarrollar un modelo conceptual para la conexión en red que fue bautizado como *Open Systems Interconnection Reference Model* o Modelo de Referencia de Interconexión de Sistemas Abiertos. Comúnmente conocida en entornos de trabajo de redes y sistemas como modelo OSI.

Este modelo que fue desarrollado por la Organización Internacional para la Normalización (ISO), pasó a ser visto como un estándar internacional en el ámbito de las comunicaciones en red, por su fácil modo de dar a entender el modo en el que los datos se desplazaban dentro de una red.

Figura 15. La pila OSI [16]

El modelo OSI está dividido en siete distintas capas. Cada una de estas etapas contiene una parte del proceso de transmisión de información entre equipos informáticos.

Las capas del modelo OSI muestran el modo en que se transmite la información dentro de una red. Sin embargo, de entre las siete capas que contiene el

modelo, únicamente dos de ellas interactúan con el usuario. Estas capas son la capa física y la capa de aplicación.

- La **capa de aplicación (capa 7)** provee la interfaz que el usuario utiliza al interactuar con la computadora, ya sea para utilizar un correo electrónico o para situar un fichero en la red.
- La **capa física (capa 1)** engloba todos los aspectos físicos o materiales de la red (tales como cables, *hubs* o cualquier otro dispositivo que se encuentre en el entorno físico de la red.

No obstante, el resto de capas no son menos importantes, puesto que cada una de las capas del modelo OSI lleva a cabo un papel esencial en la transmisión de datos de la red. En la figura 15 se puede ver representada la estructura de capas que conforman dicho modelo. Estas capas se numeran usualmente de abajo hacia arriba, aunque probablemente lo más lógico sería numerarlas de arriba hacia abajo. Éste es el sistema adoptado, pero, tanto si se utiliza el nombre como el número, lo imprescindible es recordar la función que cada una de las capas lleva a cabo dentro del proceso global de transmisión de información.

El paso previo a explicar en detalle cada una de estas capas, es necesario hacerse una idea genérica de lo que sucede cuando los datos se desplazan por el modelo OSI. La manera más sencilla de entender dicho proceso es mediante el uso de un ejemplo, un usuario desea enviar un correo electrónico a otro usuario.

El usuario emisor envía el mensaje a través de un cliente o programa de correo (como Outlook o Gmail) como instrumento de interfaz para escribir y remitir el mensaje. Como puede suponerse, esta actividad se realiza en la capa de aplicación (capa 7). Cuando la información abandona el nivel de aplicación, pasará por el resto de capas del modelo proporcionando servicios específicos, relacionados con la comunicación que debe establecerse, o bien formateando los datos de manera específica. A medida que los datos van desplazándose por las capas, cada una de ellas (a excepción de la capa física) añaden un encabezado a los datos. De esa forma llegan los datos a la capa física del receptor (entorno tangible o físico de la red, cables y hubs que conectan los ordenadores entre sí) y pasan a subir capa a capa del modelo OSI, eliminando los encabezados de los datos a medida que va pasando por cada una de ellas.

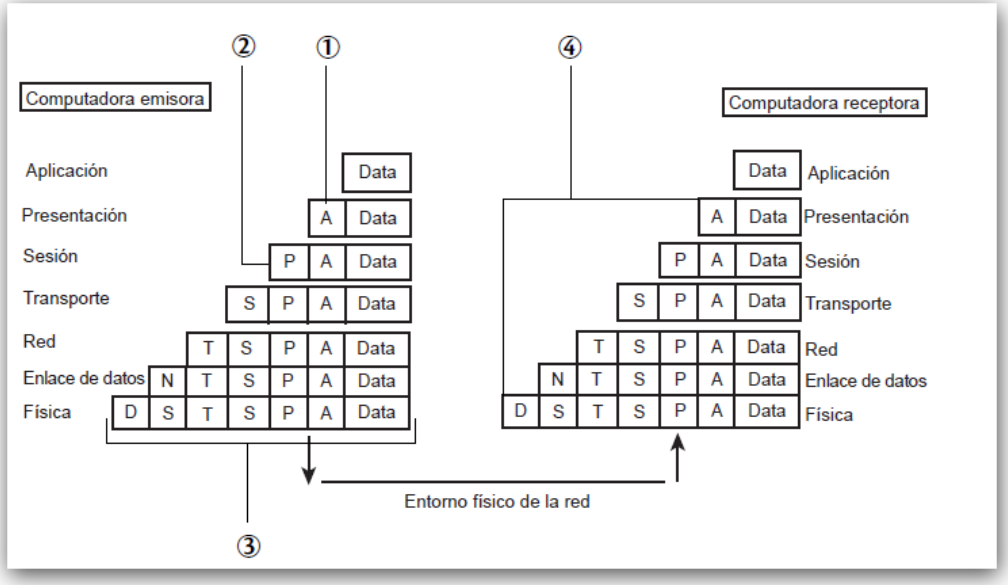

Figura 16. Transmisión de datos a través del modelo OSI [17]

Cuando la información alcanza la capa de aplicación, el receptor es capaz de leer el mensaje recibido mediante su cliente de correo electrónico.

A continuación se hará una breve explicación de cada una de las capas que conforman el modelo OSI de arriba hacia abajo.

Las capas del modelo OSI

- **La capa de aplicación (nivel 7)**

Tal y como ya se ha comentado, la capa de aplicación proporciona interfaces y servicios para la interacción con el usuario, a la vez que también es la responsable de ofrecer acceso general a la red.

Es la capa que se encarga de proporcionar al usuario las herramientas con las que actúa. Es también la capa encargada de aportar al usuario los servicios de red relacionados con estas herramientas o aplicaciones, como son la transferencia de datos, el envío y la recepción de e-mails y las consultas a bases de datos.

- **La capa de presentación (nivel 6)**

Es la capa que se encarga de la conversión de los paquetes de datos generados por la capa de aplicación a un formato genérico que sea legible por cualquier computadora.

Otra de las funciones de la capa de presentación es la de cifrar y comprimir datos para reducir su tamaño. El formato que esta capa provee es el formato en el que los datos viajarán por el resto de las capas, aunque cada una de las siguientes capas irán añadiendo datos al paquete.

- **La capa de sesión (nivel 5)**

La capa de sesión realiza el enlace de sesión o comunicación entre las computadoras emisora y receptora. Por otro lado también se encarga de la gestión de dicha sesión establecida entre ambos nodos.

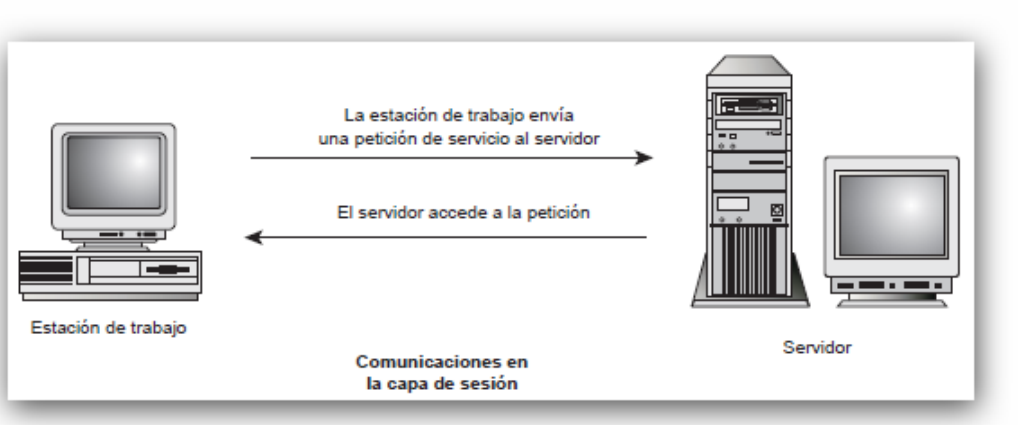

Figura 17. Funcionalidad de la capa de sesión [17]

Una vez la comunicación ha sido establecida, la capa de sesión se ocupa de situar diversos puntos de control en la secuencia de datos. De esta manera, si sucede algún error inesperado en la comunicación y esta se interrumpe, al restablecer la sesión de comunicación únicamente tendrán que enviarse los paquetes de datos que se encuentran después del último punto de control recibido, evitando así la necesidad de reenviar nuevamente todos los paquetes de datos que incluye la sesión.

- **La capa de transporte (nivel 4)**

La capa de transporte es la que se encarga de controlar el intercambio de datos entre el emisor y el receptor que han establecido una comunicación. Esta capa se encarga también de chequear que no existan errores en la secuencia de datos y además de que estos se están enviando en orden que corresponde.

Por otro lado dicha capa también se encarga de evaluar que el tamaño de los paquetes es el requerido para poder ser tratado en las capas inferiores del protocolo.

- **La capa de red (nivel 3)**

Esta capa es la que se ocupa de asignar la dirección que deben seguir los datos además de asegurarse que estos son entregados. Es la capa en la que las direcciones lógicas (tales como las direcciones IP) pasan a convertirse en direcciones físicas (como las tarjetas de interfaz de red). Y por otro lado se la capa de nivel tres se asegura que los datos son intercambiados de una manera efectiva.

- **La capa de enlace de datos (nivel 2)**

En función de la arquitectura de red que se esté utilizando, se generan diversas unidades de datos denominadas *tramas*. La capa de enlace de datos se encarga de distribuir los paquetes de datos que se reciben entre las distintas tramas.

Es importante que la capa de nivel dos asegure la recepción de las tramas a nivel de enlace físico sin ningún tipo de error. Por este motivo, en dicha capa los protocolos vigentes realizan un Chequeo de Redundancia Cíclica (*Cyclical Redundancy Check* o *CRC*) en la terminación de cada trama. Esta comprobación está basada en un cálculo numérico que se realiza tanto a nivel de nodo receptor como a nivel de nodo emisor, si ambos cálculos resultan ser el mismo valor, significará que la trama fue recibida correctamente, sin ningún error durante la transferencia.

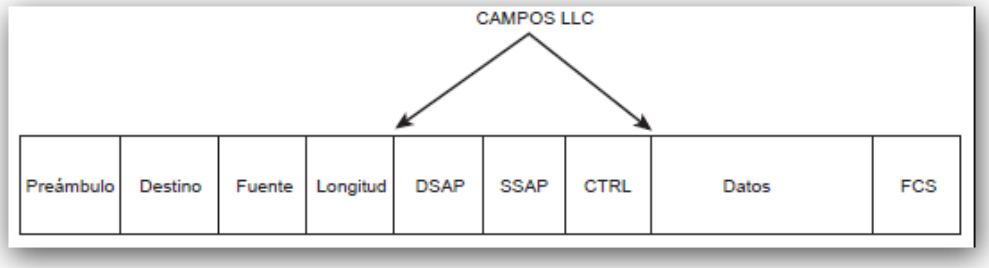

Figura 18. Ejemplo de trama dentro del protocolo Ethernet [17]

- **La capa física (nivel 1)**

La capa física se encarga de la conversión de las diversas tramas, que han llegado de la capa de nivel dos, en secuencias únicas de bits para que los datos puedan distribuirse por el entorno físico de la red. El cableado también forma parte de la tapa física.

6.3.2. Metodología MAD-SMART

La metodología MAD-SMART se basa técnicamente en dos principios básicos:

- Independencia de las técnicas de implementación
- Proceso metodológico ascendente empezando por la determinación de las necesidades del proyecto para finalizar con las estrategias de solución de este de forma cooperativa.

El modelo en cuestión consta de dos pasos diferenciados. El primero se basa en reconocimiento de posibles comunicaciones y la naturaleza de las mismas, más comúnmente conocido como "Modelo de Conocidos". El segundo paso es llevar a cabo la descripción de las posibles comunicaciones entre los agentes, más comúnmente conocido como "Modelo de Conversaciones".

En esta metodología se utilizan los diagramas de secuencia para ver las comunicaciones que se establecen entre los agentes para un escenario en cuestión. En la figura 19 se pueden ver representados algunos elementos de diagramas de secuencias con sus respectivas extensiones para agentes.

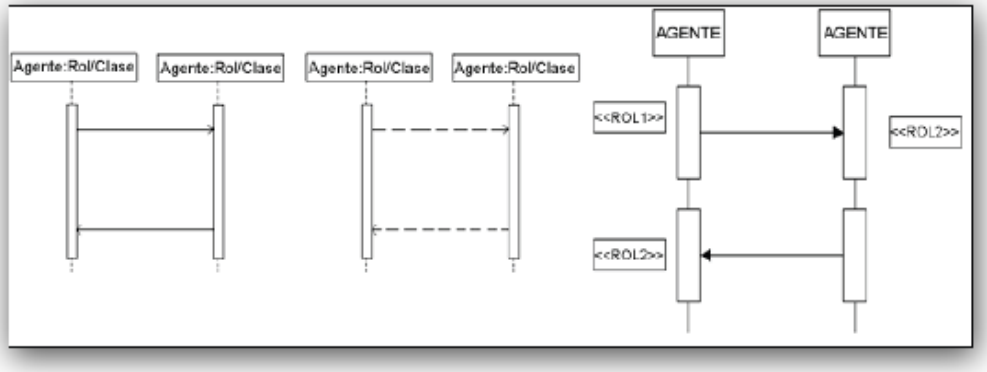

Figura 19. Comunicación directa, indicrecta y manejo de roles [18]

Por otro lado, en la figura 20 se puede observar un ejemplo de las conversaciones en un sistema de tres agentes, en un entorno de recolección de objetos. Dicha comunicación, indica que uno de los agentes encuentra un atractor y se lo comunica a los otros dos agentes. Si alguno de ellos tiene actualmente el rol de "Deambular", no entra en el escenario. Si por el contrario, ninguno de los agentes se encuentra disponible se vencerá el tiempo de espera, se cerrará la conversación y el Agente recuperará el rol "Deambular". En caso de

haber respuesta, el primer agente que haya dado señal será reclutado y ambos agentes pasarán al rol de "Entregar".

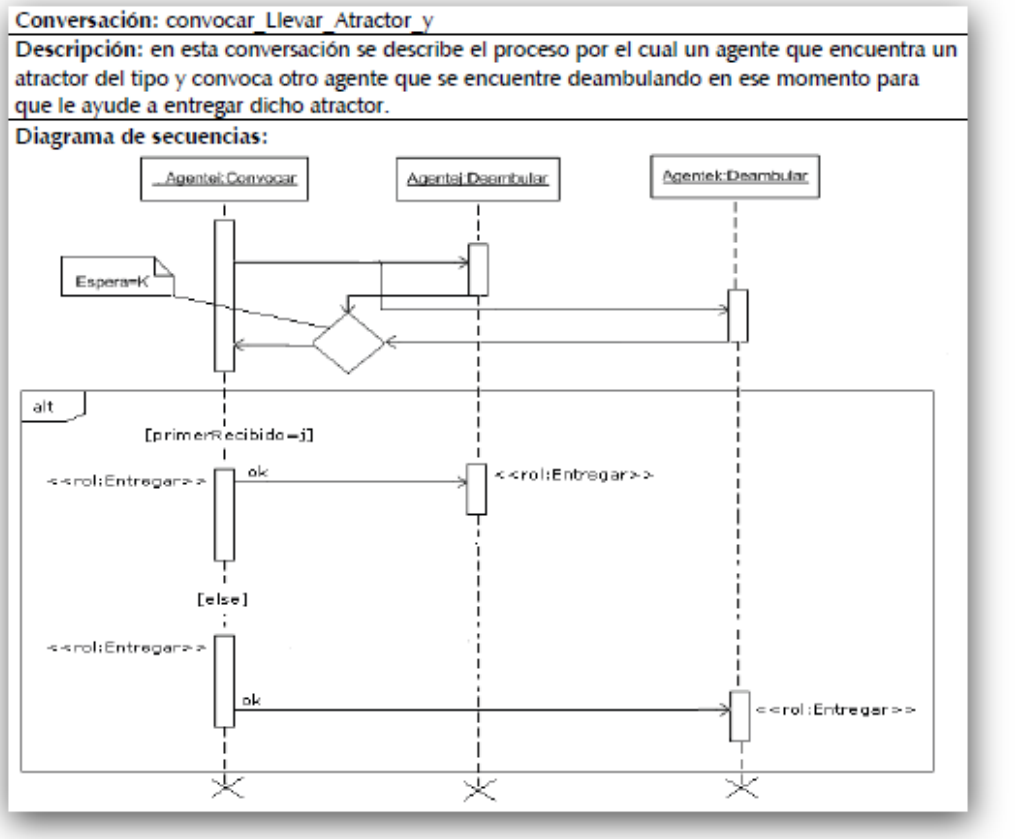

Figura 20. Conversación de agentes en un entorno de recolección de objetos [18]

En los proyectos MAD-SMART se pretende, en una primera versión, crear un entorno multi-robot para la navegación cooperativa. Todos los agentes robóticos se comunican a través de una red inalámbrica, aunque en sus versiones iniciales se planteaba el uso de comunicación por cable ya que en algunos entornos se ha mostrado igualmente útil.

En la figura 21 se puede observar la arquitectura planteada. En dicha figura se cuenta con una capa superior, en la cual están localizados los agentes robóticos inteligentes, son los que implementarán los algoritmos de navegación. Bajo esta capa se puede apreciar la pizarra del sistema, la cual hace posible el intercambio de datos entre los agentes. El entorno puede ser tanto de software, mediante simulación, como de hardware, haciendo uso de dispositivos robóticos.

Figura 21. Arquitectura del proyecto SMART [18]

Desde el punto de vista de la metodología MAD-SMART se mostrarán únicamente las conversaciones entre agente robótico y subsistema de Percepción/Comandos.

La función del subsistema Percepción/Comandos es entregar los comandos sugeridos desde la capa de agentes inteligentes a la capa de agentes robóticos y también de formato a los datos que estos agentes robóticos reciben.

6.3.3. Tecnología OPC

Previamente a realizar una explicación sobre este apartado, cabe destacar que tecnología OPC hace referencia a la capa de aplicación del modelo OSI (nivel 7). Es decir, representa una de las herramientas con las que el usuario interactúa en la red.

Concepto OPC

Un OPC, "OLE for Process Control" es un estándar de comunicación en el campo de control y supervisión de procesos industrial.

El sistema de comunicación OPC, nace en el mundo de la automatización por una sencilla razón de conectividad. A nivel de planta encontramos diversos tipos de periféricos, cada uno de ellos con un diferente tipo de conexión, tales como Ethernet, Serial o radio. Por otro lado, estos dispositivos también poseen diversos tipos de sistemas operativos, tales como Windows o UNIX.

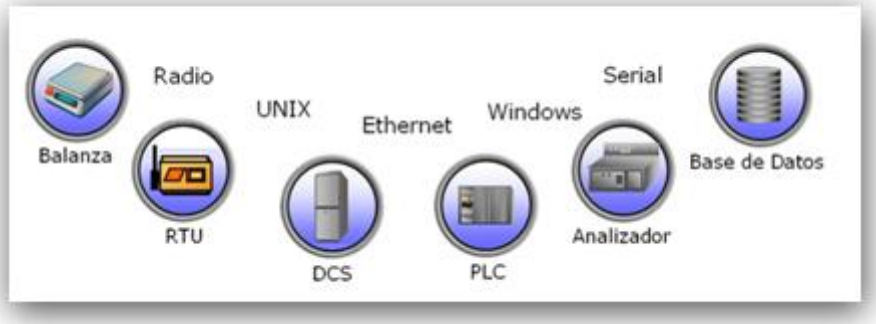

Figura 22: Tipo de periféricos con su diferente tipo de conexión [19]

Todos estos dispositivos pueden provenir de diferentes proveedores, y evidentemente, cada proveedor posee sus diferentes aplicaciones y programas. De manera que a la vez que se te vende el dispositivo físico, también se te obliga de alguna manera a comprar el software que lo mueve.

Figura 23: Dispositivos y propietarios [19]

Todos estos dispositivos de campo proveen datos, datos que vienen dados en un formato propietario. De manera que para poder utilizarlos se te obligaba utilizar las herramientas de los propietarios de dichos dispositivos. Motivo por el cual se te forzaba a dirigirte a estos proveedores si te era necesario cualquier tipo de cambio en el sistema. A raíz de todo esto, nace el concepto OPC.

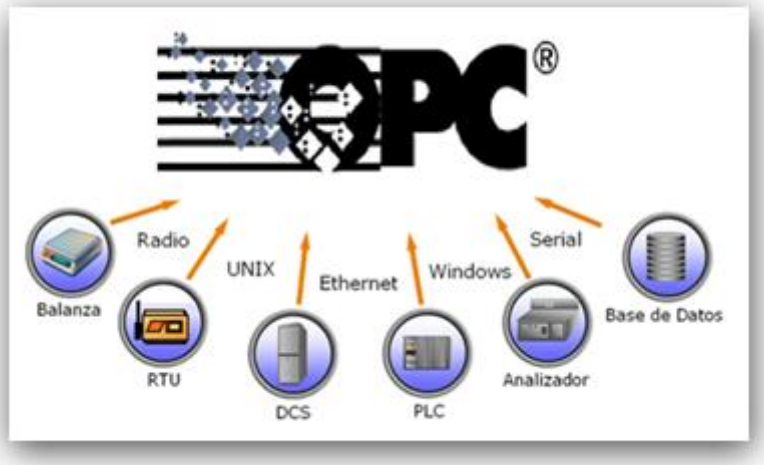

Figura 24: Concepto OPC [19]

OLE for Process Control (OPC) es un estándar de comunicación que plantea estandarizar la tecnología y no los dispositivos. Facilitando, de esta manera, el flujo de datos entre los distintos tipos de dispositivos y las posibles aplicaciones que se puedan crear para leer, o bien utilizar, estos datos. Así se hace posible utilizar dispositivos de diferentes fabricantes sin tener problemas con los diferentes tipos de datos de estos.

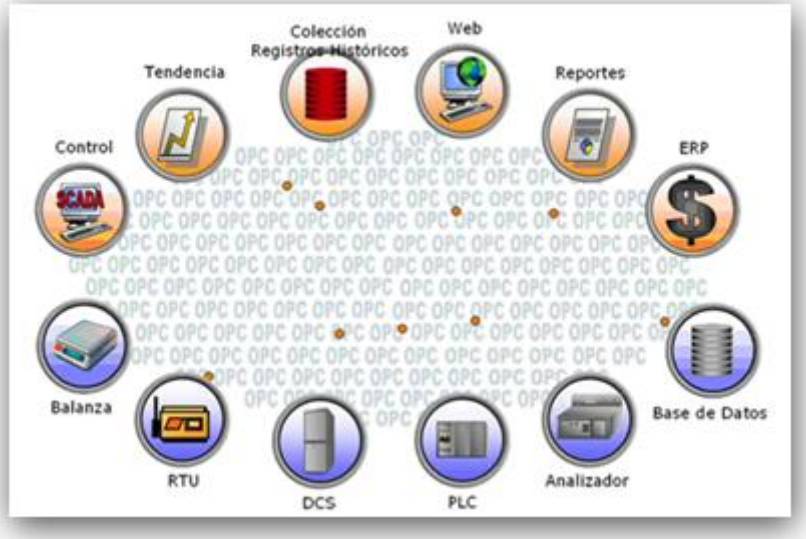

Figura 25: Visión de conectividad OPC [19]

Arquitectura OPC

La conectividad OPC se realiza utilizando una arquitectura Cliente-Servidor. Es importante recalcar que por mucho que tanto la fuente de datos como el receptor de datos puedan comunicarse entre sí, no quiere decir que sus respectivos protocolos nativos no sigan siendo necesarios. En cambio, esos protocolos nativos e interfaces siguen aún presentes, pero únicamente se comunican con uno de los dos componentes OPC. Y son estos componentes OPC los que realizan el intercambio de datos y se cierra el bucle. La información puede viajar entre la aplicación y el dispositivo sin necesidad de que establezcan comunicación entre ellos.

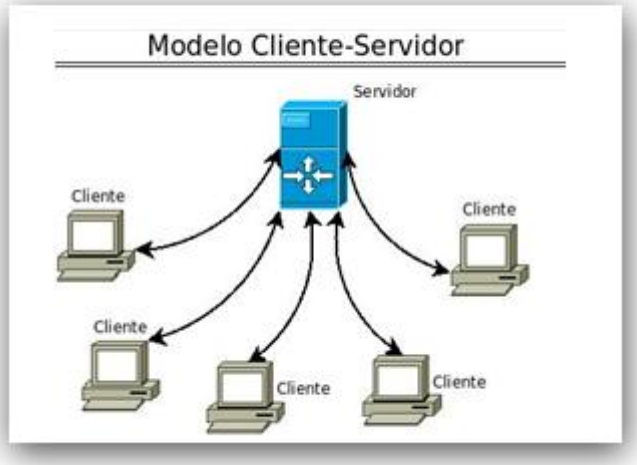

Figura 26. Arquitectura Cliente-Servidor [19]

Este método conlleva una serie de beneficios comentados a continuación:

- Una aplicación que permita la conexión OPC, puede comunicarse libremente con cualquier fuente de datos visible que permita conexión con OPC sin necesidad de utilizar ningún software o driver específico para dicha fuente de datos.

- No existe ningún tipo de limitación en el número de conexiones que pueda realizarse a una aplicación que permita conexión a OPC.

- Hoy en día OPC se ha vuelto tan conocido que prácticamente existe un conector disponible para casi cada tipo de dispositivos (modernos o antiguos) que existen en el mercado. Así que introducirse a utilizar la tecnología OPC es relativamente sencillo.

- Las fuentes de datos que permitan conexión con OPC pueden ser cambiadas, renovadas o eliminadas del sistema sin necesidad de actualizar los drivers utilizados por cualquier aplicación. Lo único que deber mantenerse actualizado siempre es el servidor OPC.

- Los usuarios pueden escoger libremente los dispositivos, controladores y aplicaciones que mejor funcionan con sus sistemas sin necesidad de preocuparse de a que vendedor pertenecen. Puesto que, provengan del vendedor que provengan, estos podrán comunicarse libremente entre ellos en cualquiera de los casos siempre y cuando sean dispositivos y aplicaciones que permitan enlace con OPC.

Servidores OPC

Un OPC server es una aplicación de software, un driver "estandarizado", creada con el fin de cumplir con una o más especificaciones OPC. La palabra "servidor" en "Servidor OPC" no se refiere al tipo de computadora en uso si no que hace referencia a la relación con su contraparte, el Cliente OPC.

Los Servidores OPC pueden ser pensados como los traductores entre el mundo OPC y cualquier protocolo nativo de comunicación o interface de una fuente de datos. Por otro lado, OPC es bidireccional, lo cual implica que los Servidores OPC pueden tanto leer como escribir en la fuente de datos. La relación entre OPC Cliente/OPC Servidor es de Maestro/Esclavo, esto significa que el Servidor OPC solo realizará una transferencia de datos hacia/desde una fuente de datos solo si un Cliente OPC lo pide.

Un servidor OPC puede conectarse con cualquier fuente de datos virtual cuya salida pueda ser leída o escrita de forma electrónica. Una pequeña lista de posibles fuentes de datos incluyen: dispositivos, PLCs, DCSs, RTUs, bases de datos, historians, páginas web y ficheros CSV que se actualicen de forma automática. Para comunicarse con cualquiera de estos elementos comentados previamente solo es necesaria la utilización de un Servidor OPC compatible con el protocolo nativo de comunicación de dicho elemento.

Aunque a nivel de usuario no es necesario que este sepa nada sobre el funcionamiento interno de un Servidor OPC para poder utilizarlo, un conocimiento conceptual de lo que está sucediendo debajo ayuda a entender porque la calidad y el rendimiento de un Servidor OPC varía en función del fabricante.

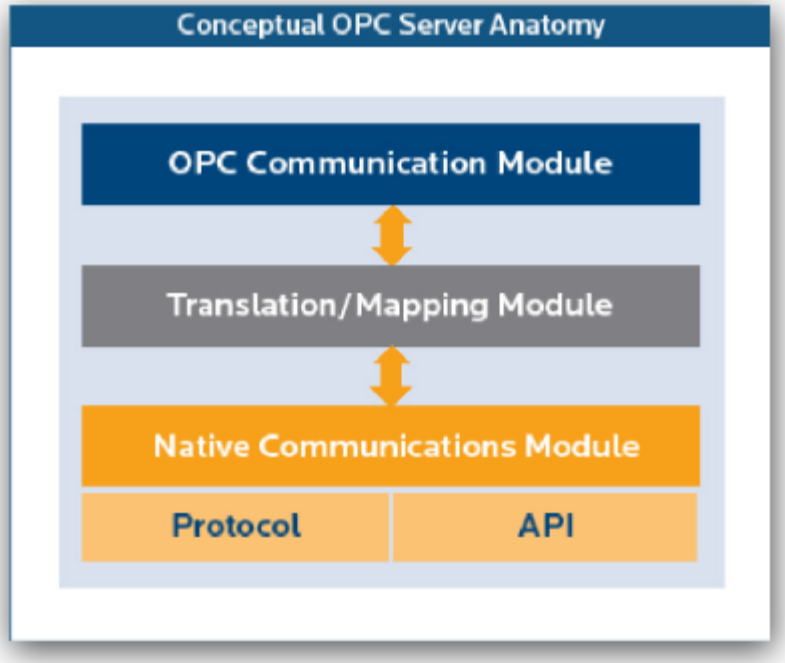

Figura 27. Anatomía conceptual de un Servidor OPC [20]

Una vista conceptual de cómo funciona interiormente un servidor OPC sería la siguiente:

- **Módulo de comunicación OPC:** Es la parte del Servidor OPC que se encarga de realizar la comunicación correctamente con un Cliente OPC dado. Un Servidor OPC bien escrito debería cumplir completamente con las especificaciones OPC que implementan para asegurar una comunicación adecuada con los Clientes OPC.

- **Módulo de comunicación nativo:** El servidor OPC debe emplear el método de comunicación más eficiente con la fuente de datos. En unos casos esto significará conectarse con la fuente de datos a través de su protocolo nativo de comunicación, mientras que en otros casos, significará establecer la comunicación con la fuente de datos a través de el driver de esta utilizando una Aplicación de Interfaz de Programación (API).

- **Módulo de traducción/mapeo:** Aquí es donde sucede toda la "magia" dentro de un Servidor OPC. Este módulo se encarga de interpretar correctamente las peticiones recibidas desde el Cliente OPC y convertirlas en peticiones nativas que serán enviadas a la fuente de datos y viceversa. Si esto se hace eficientemente, el vendedor OPC puede mantener la carga a la fuente de datos en el mínimo mientras maximiza la velocidad de transferencia de información.

Clientes OPC

Un Cliente OPC es un software escrito para comunicarse con conectores OPC. Utiliza una mensajería estipulada por las especificaciones de la Fundación OPC.

Conceptualmente, un Cliente OPC representa un receptor de datos. Este inicia y controla las comunicaciones con los Servidores OPC basado en que necesita de estos la aplicación. Dicho cliente traduce una petición de comunicación de la aplicación en una petición equivalente para que el Servidor OPC pueda interpretar la petición correctamente. Inversamente, cuando un Servidor OPC retorna los datos, el Cliente OPC debe traducir dicha información de vuelta al formato nativo de comunicación de la aplicación, para que esta pueda trabajar correctamente con los datos.

Técnicamente, un Cliente OPC es un módulo de software utilizado por una aplicación para permitir la comunicación con cualquier Servidor OPC que esté visible en su red. Usualmente es común ver los Clientes OPC dentro de otras aplicaciones como HMI's, visualizadores de tendencias, historians y escritores de reportes para hacerlos indirectamente compatibles con OPC.

De la misma manera que con los Servidores OPC, los Clientes pueden ser desglosados en tres módulos distintos en función de su zona de operación.

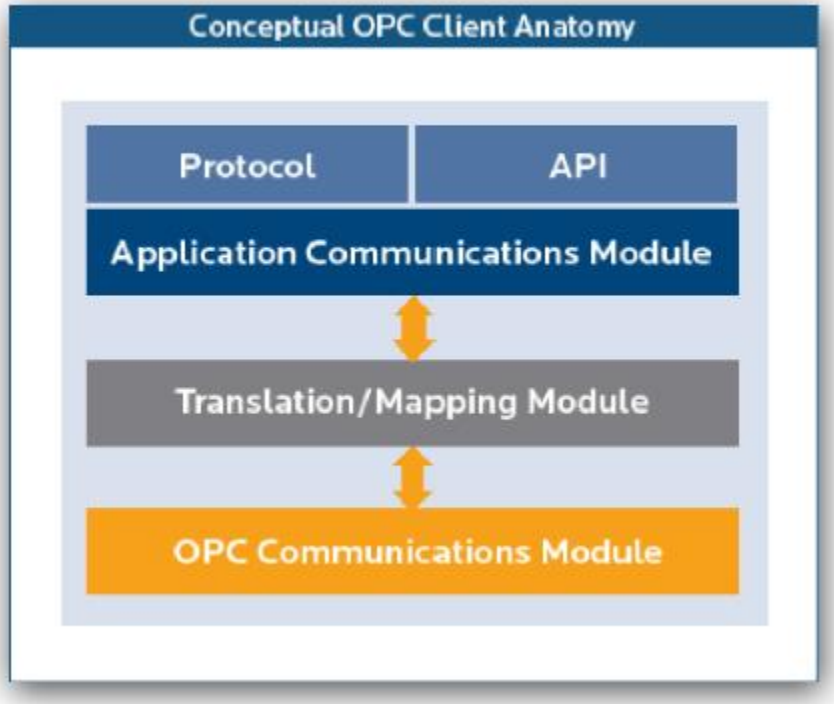

Figura 28. Anatomía conceptual de un Servidor OPC [20]

Estos tres módulos son los que se explican a continuación:

- **Modulo de comunicación OPC:** Aunque no es un módulo tan crítico como para el Servidor OPC, es igualmente crucial para el Cliente OPC comportarse correctamente mientras se conecta con el Servidor OPC, intercambiar datos con este y desconectarse sin desestabilizar al Servidor.

- **Modulo de comunicaciones con la aplicación:** El Cliente OPC está pensado para trabajar en coordinación con una aplicación específica. Así pues, el trabajo de este recae en unas cuantas peticiones por parte del API para permitir el traspaso de datos al Servidor OPC/fuente de datos a través del Cliente OPC. También es posible para un Cliente OPC genérico comunicarse con una aplicación a través de su protocolo en vez de utilizar el API si la aplicación soporta dicho protocolo.

- **Módulo de Traducción/Mapeo:** Una funcionalidad clave de los clientes OPC es que bidireccionalmente traducen información mientras que la aplicación que representan, realiza una petición de datos a leer o escribir del dispositivo o fuente de datos.

7. Planificación de caminos

7.1. Introducción a la planificación de caminos

Las aplicaciones diseñadas para robots móviles en tareas como limpieza, mantenimiento, fumigación, cosechado, exploración, fábricas, vigilancia, hogares, oficinas, etc., se encuentra en estado de auge y las únicas limitaciones que se encuentran son la necesidad de una mayor capacidad de movimiento.

Las tareas demandadas a estos autómatas van aumentando en dificultad y, últimamente, tienden a abandonar las estructuradas pruebas de laboratorio para enfrentarse a entornos cotidianos que resultan más exigentes. Así pues, la planificación de caminos resulta una tarea esencial para desarrollar dicho tipo de misiones.

La planificación de tareas consiste en decidir el orden de las acciones a realizar para lograr su ejecución en un determinado modelo de entorno.

Se sabe que se ha realizado una correcta planificación de caminos cuando, tal y como se comentó en el tema anterior, el robot es capaz de responder a dos preguntas básicas:

- **¿Dónde estoy?:** Hace referencia a la necesidad de localizarse dentro de un mapa conociendo los obstáculos móviles que le rodean y los puntos fijos en base a los cuales hará referencia para mantener la localización con una precisión bastante elevada.
- **¿Cómo es el mundo que me rodea?:** Debido a la imprecisión de los sensores odométricos, que es la primera fuente de información de la cual hacen uso los robots con el fin de conocer cuál es su posición, surge la necesidad de utilizar los puntos fijos como referencia.

7.2. Problemática en la planificación de camino

Generalizando, un robot resulta ser una herramienta muy versátil debido a su fácil adaptación a los diferentes entornos. Puede poseer más o menos sensores o actuadores y tienen también diferentes formas, siempre en función de la tarea que deban desarrollar. Estas tareas, se llevan a cabo en un determinado espacio de trabajo. Dicho espacio es un espacio limitado, y por lo general estará ocupado por otros objetos. Evidentemente, lo deseable sería que este robot estuviese integrado completamente en el espacio que opera, de forma que pudiese operar si tener problemas con los demás elementos del entorno.

Por otro lado, está el hecho de que la planificación de caminos no es un problema bien sencillo y bien definido, sino que este se trata de un conjunto de diversos problemas a tratar. Por lo que no es sencillo encontrar una solución al problema que pueda cubrir efectivamente cualquier casuística.

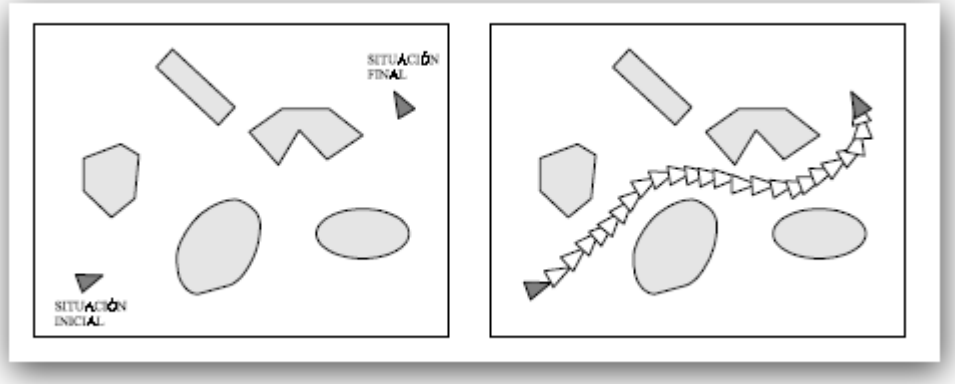

Figura 29. Problema básico de planificación de movimientos [23]

Eludiendo el hecho de que hay diversos problemas a tratar por separado. Se podría formular un enunciado generalista tratando de englobar todos esos problemas. Obviamente, el problema básico cuenta con una gran simplificación. No obstante, la navegación de un robot móvil es un caso aproximadamente real del problema básico. Es decir, un estudio inicial de dicho problema haría posible extrapolar la solución a situaciones reales.

El enunciado del problema básico puede ser definido de diferentes maneras. Por ejemplo, citando íntegramente desde [24].

'Dado un objeto con una configuración (posición y orientación) inicial, una configuración destino, y un conjunto de obstáculos distribuidos en el espacio, encontrar un camino continuo para el objeto desde la configuración inicial hasta

la configuración destino sin colisionar con ningún obstáculo a lo largo del camino.'

La figura 29 hace referencia a dicho enunciado. No obstante, tal y como se ha comentado, la resolución de dicho problema no podría considerarse suficientemente fiable. Por lo que a continuación se comentan brevemente hacia donde podría extenderse dicho problema.

7.2.1. Múltiples objetos móviles

Podríamos incluir tres ampliaciones del problema bajo este epígrafe, que eliminarían algunas de las suposiciones necesarias para poder realizar el problema simple.

- Al existir limitaciones por tiempo y aceleración, se debe traducir en restricciones geométricas en la pendiente y la curvatura de la trayectoria que coja el robot en función del tiempo.
- El uso simultáneo de diversos robots en el mismo entorno. Los otros robots pasan a ser considerados un obstáculo potencial entre sí. Este problema puede enfocarse desde diferentes puntos de vista.
    - Considerar todos los robots como si fueran un único robot de varios cuerpos (*single bodied robot)*
    - Planificar, en primera fase, la trayectoria de forma separada para cada uno de los robots. Mientras que en una segunda fase, se consideraría la intersección entre los caminos de estos.
    - Planificar, de forma secuencial, la programación de cada uno de los robots y pasar a considerar el resto de ellos como si de otros objetos móviles se tratase.
- En robots articulados, cada una de las partes del robot está sujeta a las restricciones que le aplican las otras partes en contacto con ella. El mayor problema que se presenta en este caso consiste en que el número posible de configuraciones aumenta en función de la cantidad de articulaciones.

7.2.2. Restricciones cinemáticas

En la resolución del problema básico, únicamente se aplica la restricción de la posible presencia de obstáculos en el camino del robot. Mientras que en la práctica existen otras restricciones cinemáticas al movimiento del robot llamadas restricciones holonómicas o no-holonómicas.

Una restricción holonómica consiste en una configuración que es representada por una lista de parámetros de cardinal mínimo. Resulta ser una relación de igualdad entre estos parámetros que reduce la dimensión actual del espacio de configuraciones. Restringiendo las dimensiones en las cual el robot puede moverse. Por otro lado, una restricción no-holonómica consiste en una ecuación no integrable incluyendo tanto la lista de parámetros que representa la configuración como sus derivadas. Mientras que la restricción holonómica restringía las dimensiones en las cual el robot podía actuar, esta otra restringe las dimensiones de la velocidad.

7.2.3. Incertidumbre y objetos móviles

No es posible asumir que se conoce de una manera cien por cien fiable tanto la posición del robot como la de los obstáculos. De igual manera, tampoco es posible asumir que los robots reproducen fielmente el camino generado por el planificador. No obstante, estos pequeños errores no son realmente importantes teniendo en cuenta la tolerancia que se le aplica al movimiento del robot.

En un caso extremo el robot podría confiar en sus sensores de actuación durante la ejecución de modo que sin tener un conocimiento profundo del entorno. Sin embargo, cuanto más se aleje de la realidad la estimación inicial del entorno. Mayores serán los errores que se cometan a la hora de realizar el camino propuesto por el planificador de movimientos.

Previamente se ha hablado de la posibilidad de que existan otros objetos o robots moviéndose por el espacio. Pero ahora se hace referencia a la existencia de objetos cuya posición pueda ser modificada por el propio robot, de dicha manera pasarían a existir nuevos caminos o configuraciones en los cuales el robot pasaría a desplazarse en conjunción con dichos objetos. El principal problema de esto reside en que espacio puede utilizarse, es decir, en qué momento el robot debe pasar de la configuración conjunta con el objeto móvil a la configuración de movimiento solo.

7.3. Métodos de planificación de trayectorias

7.3.1. Grafos de visibilidad

Este método de generación de grafos se basa en el término visibilidad para realizar los grafos. Si existe la posibilidad de unir dos puntos, en un tramo rectilíneo, sin ser interceptado por ningún tipo de obstáculo, estos puntos pueden ser considerados *"visibles"*. En este tipo de grafos pueden considerarse como nodos el punto de inicio, el punto final y todos los vértices de los objetos que se encuentren dentro del entorno. Siendo el grafo resultante la unión de todos los nodos visibles, se muestra un ejemplo de grafo en la figura 30.

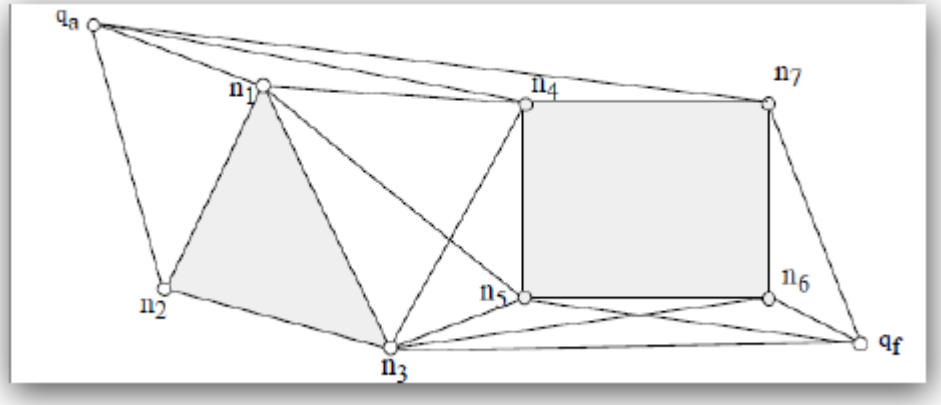

Figura 30. Ejemplo de grafo de visibilidad [25]

Posteriormente, con la ayuda de algún algoritmo de búsqueda de grafos se escoge la ruta óptima.

7.3.2. Diagramas de Voronoi

Se trata de una de las estructuras más potentes en geometría computacional, su principal. Estos diagramas consisten en una proyección del espacio libre en una red curvas unidimensionales que yacen en el mismo espacio.

La idea principal del diagrama de Voronoi reside en maximizar la distancia existente entre los robots y los obstáculos. Por lo que el diagrama de Voronoi resulta el lugar geométrico de esas configuraciones equidistantes a los obstáculos más cercanos del entorno (Vease la figura 31)

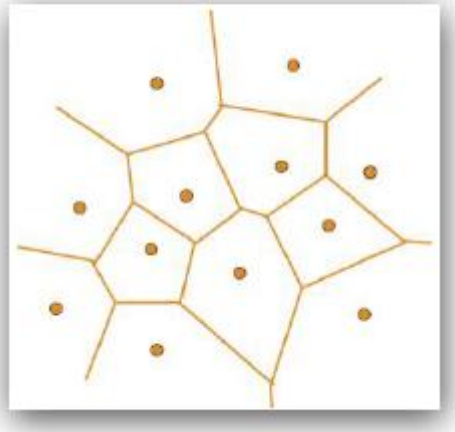

Figura 31. Ejemplo de diagrama de Voronoi [25]

Para una representación más realística los obstáculos pasan a considerarse como polígonos. Puesto que técnicamente, los obstáculos no son simplemente puntos. Así pues, el diagrama de Voronoi pasará a estar formado únicamente por curvas y rectas, tal y como se representa en la figura 32. Donde las líneas gruesas representan la solución de dicho diagrama.

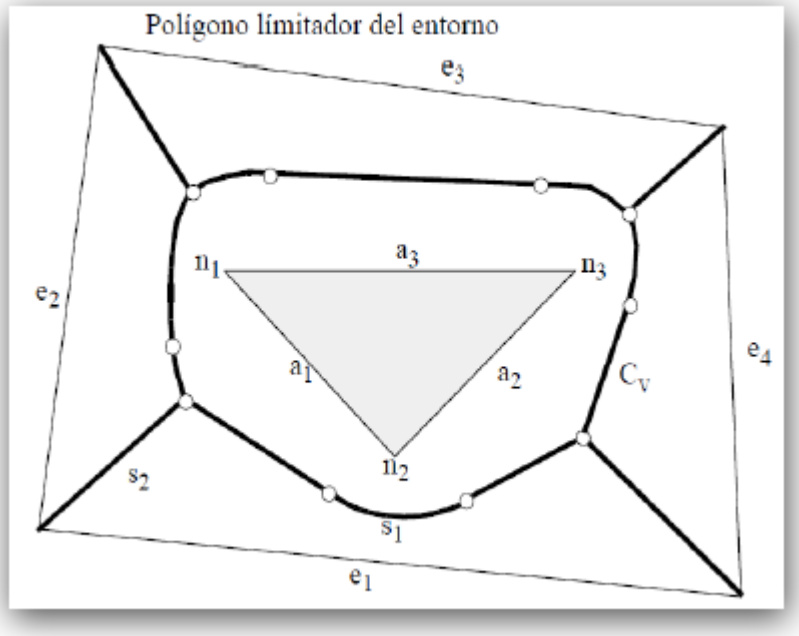

Figura 32. Diagrama de Voronoi, retracción del espacio libre [25]

Este método también trabaja dentro de entornos conocidos y obstáculos poligonales. No obstante, también se pueden encontrar algunas versiones del Diagrama de Voronoi trabajando con obstáculos móviles e inesperados.

7.3.3. Roadmap Probabilístico (PRM)

Este método se basa en la creación de un número n de configuraciones aleatorias sin posibilidad de colisión alrededor de todo el entorno.

Posteriormente se realiza una conexión de todos los nodos en función de la proximidad entre ellos, dependiendo del número de objetos existentes en el área de trabajo. Y una vez finalizado este proceso, se ejecuta un algoritmo para obtener la ruta óptima desde un nodo inicial a un nodo final.

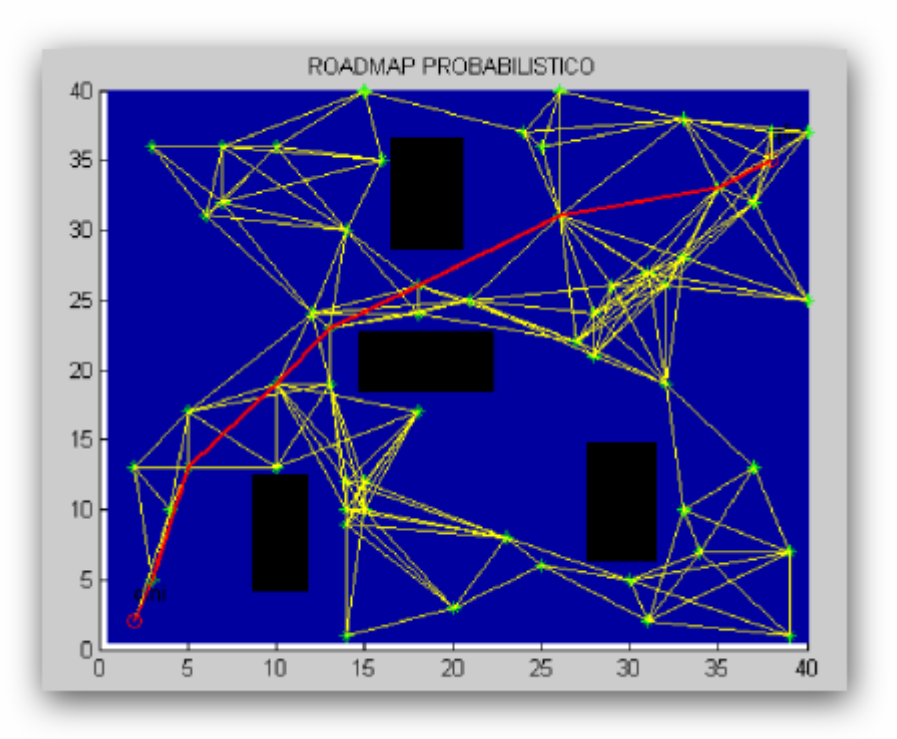

Figura 33. Roadmap Probabilístico [25]

7.3.4. Modelo del espacio libre

En este caso la planificación se lleva a cabo a través de un método llamado cilindros rectilíneos generalizados (CRG). Este método también pretende hacer que los robots circulen lo más lejos posible de cualquier tipo de obstáculo.

La construcción de un CRG puede verse en la figura 34.

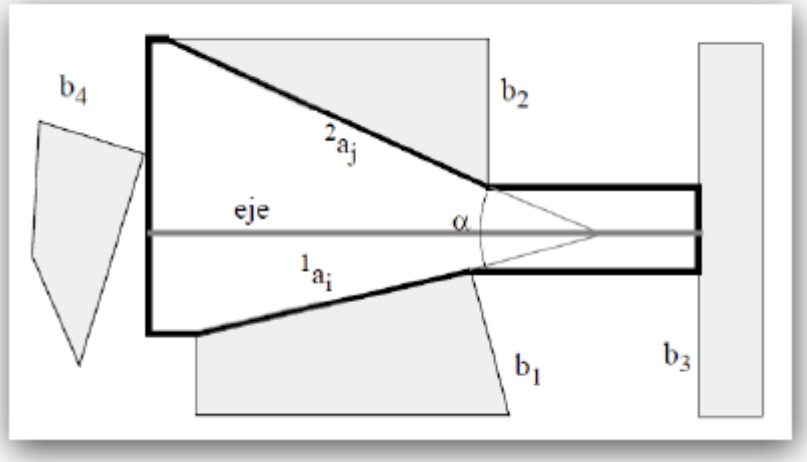

Figura 34 Construcción de un CRG [25]

Este método se realiza siguiente una serie de cálculos de CRG con el objetivo de crear una red de CRG's que modelan el espacio libre por el que circulará el robot. El paso de un CRG a otro se producirá siempre y cuando sus ejes se intercepten y la intersección del rango de orientaciones admisibles en el punto de corte para ambos ejes no sea nula.

7.4. Algoritmos de planificación de caminos

En este apartado se realiza una explicación generalista de diversos algoritmos utilizados en la planificación de caminos.

Se realizará únicamente un ejemplo de funcionamiento del algoritmo de Dijkstra, ya que es el método que se ha seleccionado para realizar la planificación de caminos en este proyecto.

7.4.1. SLAM (Simultaneous Localization and Mapping)

El método SLAM (Localización y Mapeo Simultáneos) es un objeto que utiliza la arquitectura cliente servidor cuyos objetivos son:

- Realizar una estimación precisa de la posición del robot para cualquier instante de tiempo
- Reproducir un mapa actualizado del entorno

Para poder llevar a cabo dichos objetivos este módulo requiere la entrada de datos procedentes de otros dos objetos tal y como se muestra en la figura 35.

- Por un lado, reclama la información que proviene de un dispositivo laser y las rectas que identifican el entorno
- Por otro lado, recibe los datos odométricos a través de unos sensores propios del robot.

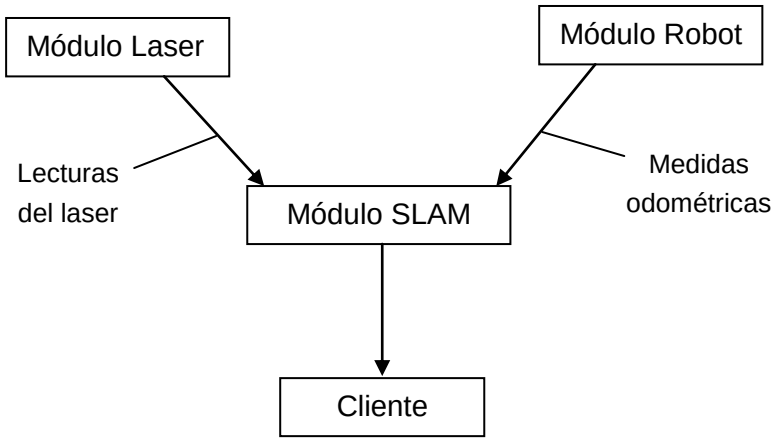

Figura 35. Relación entre módulos SLAM

7.4.2. Algoritmo buscador de caminos de Floyd-Warshall

En ciencia de la computación, el algoritmo de Floyd-Warshal (también conocido como algoritmo de Floyd, algoritmo de Roy-Warshall, algoritmo de Roy-Floyd o algoritmo WFI) es un algoritmo de análisis gráfico para encontrar el camino más corto en un grafo con pesos, tanto positivos como negativos. Una simple ejecución del algoritmo encontrará las longitudes (pesos sumados) de los caminos más cortos entre todas las parejas de vértices del grafo, pero no devuelve detalles del camino en sí.

El algoritmo de Floyd-Warshall fue publicado en su forma públicamente reconocida por Robert Floyd en 1962. Sin embargo, es esencialmente el mismo algoritmo previamente publicado por Bernard Roy en 1959 y también por Stephen Warshall en 1962 para encontrar el cierre transitivo de un grafo. El algoritmo es un ejemplo de programación dinámica.

Es importante remarcar que pueden suceder ciclos negativos. Un ciclo negativo es un ciclo cuyos ejes suman un valor negativo. No hay camino más corto entre ninguna pareja de vértices i,j que forman parte de dicho ciclo negativo, porque las longitudes de los caminos desde i hasta j pueden ser arbitrariamente pequeñas (negativas). En referencia a la salida, numéricamente hablando, el algoritmo de Floyd-Warshall asume que no hay ciclos negativos. No obstante, si hay ciclos negativos, el algoritmo puede ser usado para detectarlos. Una vez se ha detectado uno de estos ciclos, uno puede chequear dicho grafo y si este contiene una diagonal negativa, este tendrá al menos un ciclo negativo. Obviamente, en un grafo indirecto, un eje negativo crea ciclos negativos involucrando los vértices adyacentes.

Entre otras cosas, el algoritmo de Floyd-Warshall puede ser utilizado para resolver los siguientes problemas, entre otros:

- Camino mínimo en grafos directos.
- Cierre transitivo de grafos directos.
- Encontrar la expresión regular denotando el lenguaje ordinario aceptado por una automatización finita.
- Inversión de matrices reales
- Encontrar rutas óptimas, en este caso en vez de tratar de encontrar las rutas con mínimo peso, se tratan de encontrar las de máximo peso.
- Computación rápida de redes de buscadores de caminos
- Cálculo de la forma canónica de la diferencia de matrices ligadas.

7.4.3. Campos potenciales

Este método pasa a considerar al robot como si de una partícula bajo la influencia de un campo potencial artificial "U" se tratase.

Dicho campo potencial "U", se define como la unión de dos componentes. El componente atractivo, el cual atrae al robot hacia la configuración o posición final. Y el componente repulsivo, el cual repele al robot fuera del alcance de los obstáculos.

El método de los campos potenciales es un algoritmo iterativo que itera en base a la siguiente función:

$$U(q) = Uatrac(q) + Urep(q)$$

Donde "Uatrac(q)" representa la componente atractiva y "Urep(q)" representa la componente repulsiva. El efecto que este sistema haría es algo parecido a lo que se muestra en la figura 36.

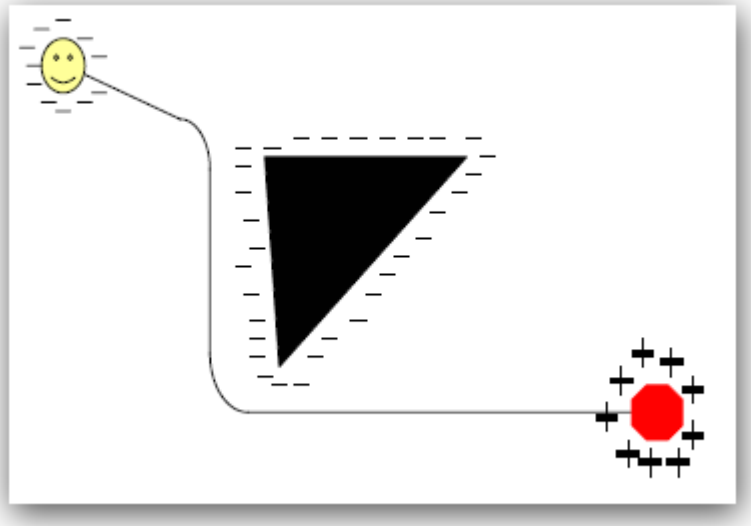

Figura 36. Efecto del campo potencial sobre el robot [26]

Se puede apreciar la forma aproximada que representarían la componente atractiva y la componente repulsiva en las figuras 37 y 38 respectivamente.

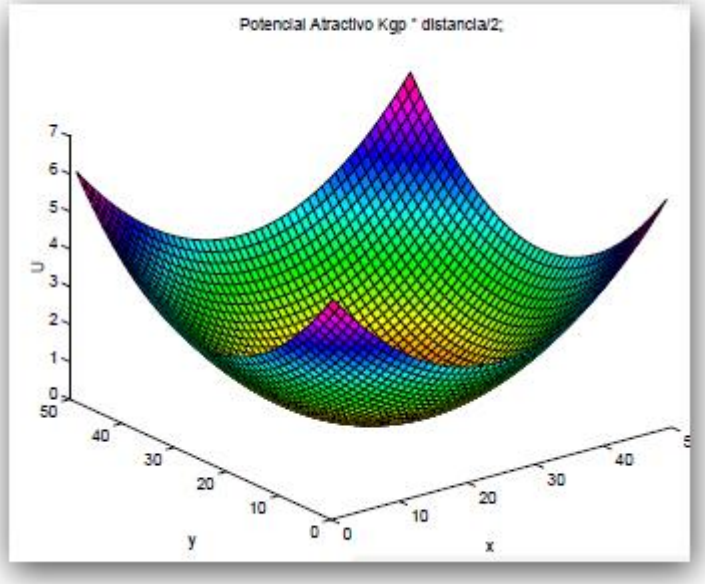

Figura 37. Representación de la componente atractiva del campo potencial [25]

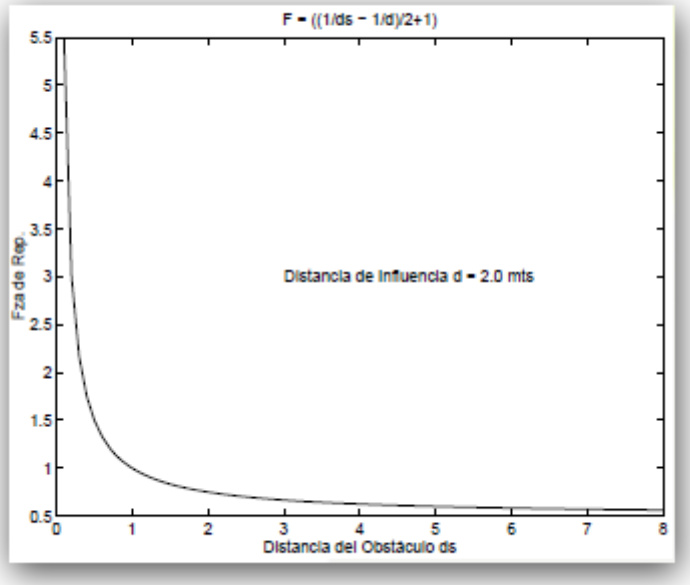

Figura 38. Representación de la componente repulsiva del campo potencial [28]

Un problema que aparece con frecuencia en este tipo de modelos es la presencia de mínimos locales. Estos mínimos locales aparecen en los puntos en que ambas componentes (atractiva y repulsiva) se igualan. Cuando esto sucede, pueden ocurrir dos casuísticas distintas. Una de ellas resultaría en el bloqueo del movimiento del robot, la otra resultaría en la permanencia del robot en una determinada área debido a la imposibilidad de salir de este. Un ejemplo de dicha casuística se muestra en la figura 39.

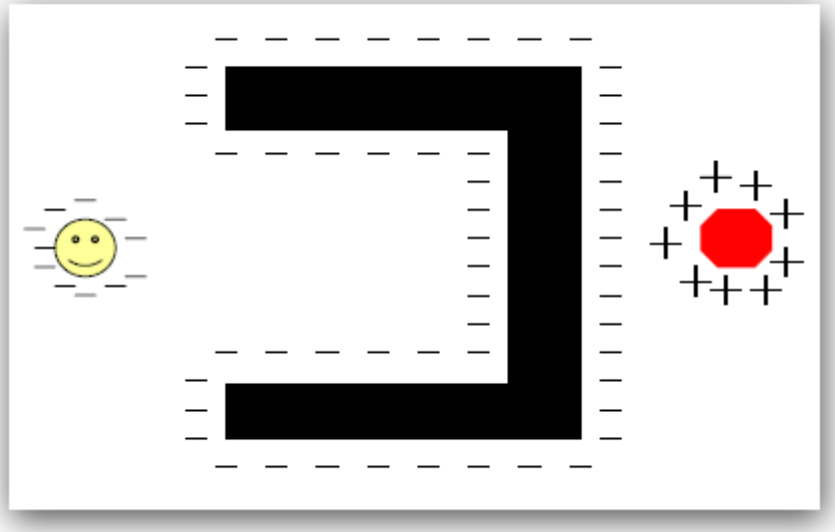

Figura 39. Problemática de los mínimos locales [28]

7.4.4. Algoritmo Dijkstra

Este sistema se basa en un algoritmo que permite, dado un grafo como el que se muestra en la figura 1, determinar el camino óptimo que une dos nodos. Donde los nodos podrían representar entradas, salidas, almacenes, puestos de producción, etc.

Puede comprobarse que entre nodo y nodo hay unos pesos aplicados. Estos pesos representan el grado de dificultad hallada para moverse entre nodos, este grado de dificultad puede venir dado por distancia, por impedimentos del camino, etc.

El algoritmo hace un barrido completo de todo el grafo independientemente de de los nodos de inicio y de fin, descubriendo así el peso óptimo para alcanzar cada nodo desde el marcado como nodo de inicio.

A modo de figura, se muestra a continuación un ejemplo de implementación de dijkstra, para un más sencillo entendimiento del mismo.

Pequeño ejemplo didáctico de Dijkstra

Previamente hay que dejar claro que los nodos podrán tener dos tipos de propiedades, "PesoAcumulado" y "NodoAntecesor".

"PesoAcumulado" corresponde a la suma de pesos entre nodos. En el ejemplo mostrado a continuación se muestra en diversas ocasiones.

"NodoAntecesor" corresponde al nodo que precede al nodo que ahora mismo se está estudiando con menor peso acumulado.

A efectos de la explicación se utilizará el grafo que se muestra en la siguiente figura, y se buscará el camino óptimo para desplazarse desde el nodo 1 hasta el nodo 5:

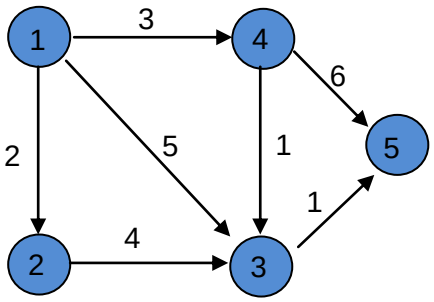

Figura 40. Ejemplo de grafo

- **Primera iteración.** Se selecciona un nodo de inicio, en este caso será el nodo 1, y se iniciaran las dos variables de las cuales se ha hablado antes.

  PesoAcumulado = 0; (puesto que acaba de empezar la iteración)

  NodoAntecesor = Nulo; (puesto que no le precede ningún nodo)

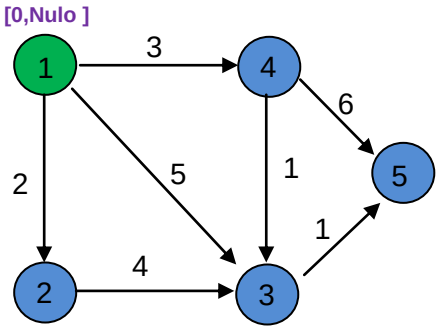

Figura 41.Primera iteración del algoritmo Dijkstra

Como puede comprobarse en la figura anterior, se ha marcado con color verde el nodo inicial, y consecuentemente se marcarán también los nodos se hayan tomado como nodo central. Y por consiguiente estos nodos no volverán a ser visitados.

Por otro lado se marcará en el ejemplo (de color lila) las dos propiedades comentadas con anterioridad. Siguiendo el siguiente patrón: [NodoAntecesor, PesoAcumulado].

- **Segunda iteración.** Se marcarán todos los nodos adyacentes al nodo 1 inicializando sus variables siguiendo el siguiente patrón:

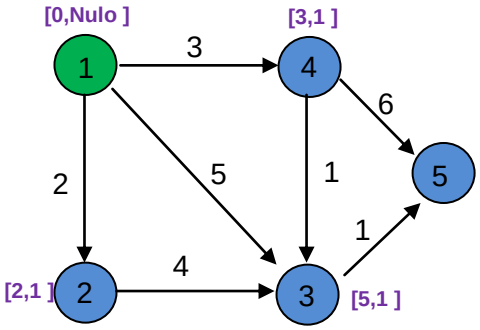

Figura 42. Segunda iteración del algoritmo Dijkstra

PesoAcumulado = [PesoAcumulado del nodo anterior(en este caso el nodo 1)] + [peso de la arista que los une]

NodoAntecesor = el nodo del cual proviene.

De esta forma:

Para nodo 2

PesoAcumulado(nodo 2) = 0 + 2 = 2

NodoAntecesor = 1

Para nodo 3

PesoAcumulado (nodo 3) = 0 + 5 = 5

NodoAntecesor = 1

Para nodo 4

PesoAcumulado(nodo 4) = 0 + 3 = 3

NodoAntecesor = 1

- **Tercera iteración.** En esta ocasión, el nodo con menor peso acumulado es el nodo 2. Así que se repetirá el proceso sin volver a comprobar los nodos que ya se han utilizado previamente (en verde).

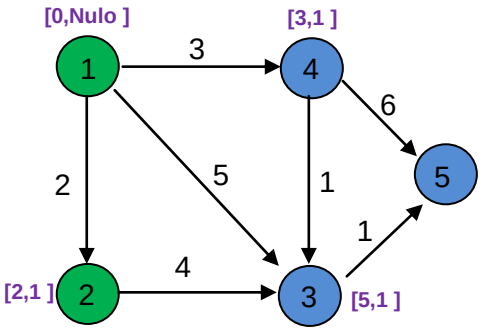

Figura 43. Tercera iteración del algoritmo Dijkstra

Para nodo 3

newPesoAcumulado (nodo 3) =2 + 4 = 6

PesoAcumulado(nodo3) < newPesoAcumulado(nodo3)

PesoAcumulado(nodo3) = 5

NodoAntecesor = 1

Para nodo 4

newPesoAcumulado(nodo4) = 2 + infinito = infinito

PesoAcumulado(nodo4) < newPesoAcumulado(nodo4)

PesoAcumulado(nodo4) = 3

NodoAntecesor = 1

- **Cuarta iteración.** En este caso el nodo 4 es el nodo de menor peso acumulado, por lo que se centra el proceso en él. (se pinta de verde para señalar que ya se ha comprobado)

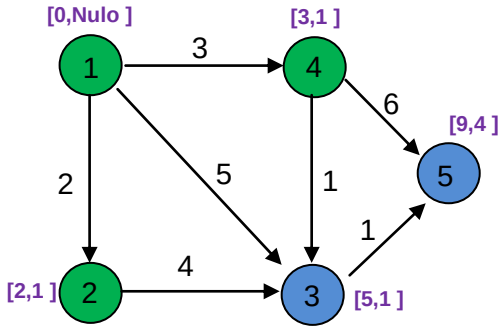

Figura 44. Cuarta iteración del algoritmo Dijkstra

Para nodo 3

newPesoAcumulado (nodo 3) =3 +1 = 4

PesoAcumulado(nodo3) > newPesoAcumulado(nodo3)

PesoAcumulado(nodo3) = newPesoAcumulado(nodo3) = 4

NodoAntecesor = 4

Para nodo 5

PesoAcumulado(nodo5) = 3 + 6 = 9

NodoAntecesor = 4

- **Quinta iteración.** En este caso el nodo 3 es el nodo de menor peso acumulado, por lo que ahora se centra el proceso en él. (se pinta de verde para señalar que ya se ha comprobado)

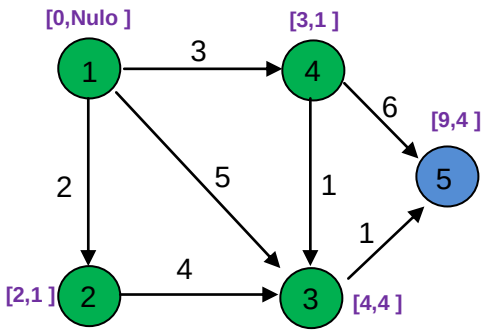

Figura 45. Quinta iteración del algoritmo Dijkstra

Para nodo 5

newPesoAcumulado (nodo 5) =4 +1 = 5

PesoAcumulado(nodo5) > newPesoAcumulado(nodo5)

PesoAcumulado(nodo5) = newPesoAcumulado(nodo5) = 5

NodoAntecesor = 3

- ***Solución***

De este modo se da por finalizada la exploración del grafo, puesto que todos los nodos han sido comprobados.

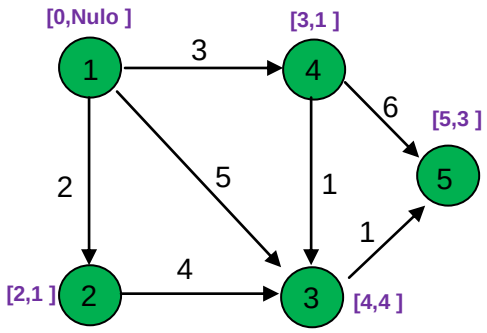

Figura 46. Solución del ejemplo Dijkstra

En la situación del grafo del ejemplo, el coste total òptimo en trasladarse des del nodo 1 al nodo 5 es de 5. Y el camino óptimo teniendo en cuenta el nodo antecesor en cada situación es 1-4-3-5.

8. Supervisión de procesos

8.1. Introducción a la supervisión de procesos

Cada vez existen más procesos productivos automatizados en el sector de la industria. Estos procesos incluyen una gran diversidad de elementos de distintas complejidades: ordenadores, autómatas, robots, etc. Además, a medida que avanza el tiempo aparecen nuevos sistemas de producción más flexibles que los que se utilizaban con antigüedad. Anteriormente se utilizaban sistemas de producción fuertemente centralizados y con muy poca flexibilidad, dichos sistemas ya no pueden permitirse puesto que no son capaces de adaptarse a la fuerte variabilidad a la que está sometido este mercado. Por tanto, cuanto más aumenta la flexibilidad de los sistemas, más dificultad existe a la hora de realizar el control que lo gobierne.

Todo esto ha dado paso a la creación de sistemas de control más inteligentes que se basan en: autonomía, cooperación y colaboración, monitorización, etc. Debido a la creación de estos sistemas, es necesario que el personal que se encarga del mantenimiento de los mismos sean altamente cualificados y con una continua formación.

Siguiendo con esta línea de investigación se pretende potenciar el campo de control y supervisión de sistemas complejos basados en redes y buses de campo. Profundizando sobre todo en aquellos sistemas que combinan diferentes tecnologías, de modo que la comunicación entre todos los dispositivos que entran en juego sea el máximo de transparente y eficaz posible.

8.2. Los Sistemas SCADA

8.2.1. Introducción a los sistemas SCADA

SCADA es el acrónimo de Supervisory Control And Data Acquisition (control y adquisición de datos de supervisión). Este es un sistema industrial de cuyas principales tareas son monitorizar y controlar procesos industriales. Los SCADA constan de un ordenador principal o master (al que generalmente se le llama MTU, "*Master Terminal Unit*" o Estación principal), una o más unidades de control que proveen datos de campo (denominadas usualmente RTU's, "*Remote Terminal Units*" o estaciones remotas) y por último cuentan con una cantidad importante de software estándar hecho a medida para controlar y monitorizar de forma remota cualquier dispositivo de campo a través de un HMI ("*Human Machine Interface*"). Los sistemas SCADA muestran características de control a lazo abierto y utilizan comunicaciones interurbanas. No obstante, también están

presentes algunos elementos de control a lazo cerrado con comunicaciones de corta distancia.

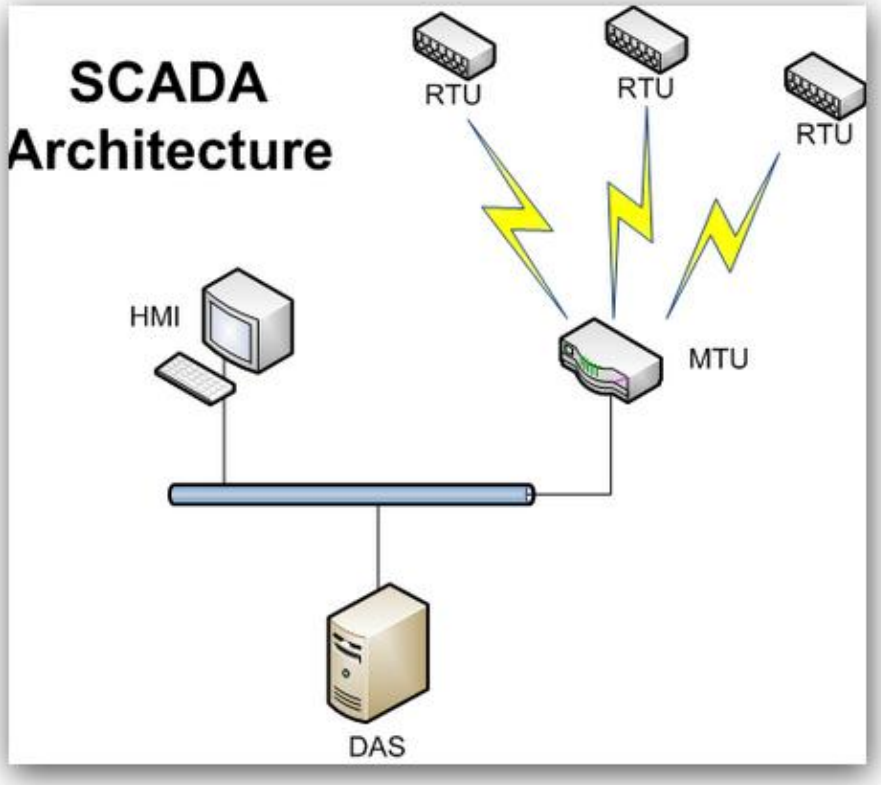

Figura 47. Arquitectura SCADA [32]

Los primeros sistemas SCADA que aparecieron, eran simples sistemas de telemetría que facilitaban reportes periódicos de las condiciones de campo vigilando las señales que representaban y las condiciones de estado en ubicaciones de campo remotas. Estos primeros sistemas SCADA fueron modificados en gran medida por las aplicaciones específicas para cumplir los requisitos de algún proyecto en particular. Ingenieros de diversas industrias asistieron a la presentación de dichos sistemas, es por eso que la percepción de los sistemas SCADA de cada uno vino condicionada por las características de su propia industria. Únicamente cuando nuevos proyectos pidieron requisitos y aplicaciones adicionales, hizo que los desarrolladores SCADA adquiriesen experiencia trabajando junto con otras industrias.

Comúnmente se pueden encontrar sistemas parecidos a SCADA en fábricas, plantas de tratamiento, etc. Dichos sistemas son nombrados como Sistemas de Control Distribuidos (DCS, "*Distributed Control Systems"*). Su funcionalidad es similar a la de SCADA, con la excepción de que las unidades de recolección y control de datos se instauran normalmente en el interior de una área confinada.

Por tanto, las comunicaciones que se establecen se realizan generalmente mediante una red de área local (LAN, "*Local Area Network*"), dicha comunicación usualmente garantiza confianza y una velocidad alta. Por otro lado, un sistema SCADA por lo general cubre áreas más extensas, siendo dependiente de una gran variedad de sistemas de comunicación menos fiables que una red LAN.

Se puede generalizar que SCADA, es una herramienta utilizada para monitorizar y controlar una planta industrial o equipamiento, cuyo control puede ser o bien automático, o iniciado por comandos de operador. La toma de datos se realiza principalmente por los RTU's que rastrean las entradas de información de campo que están conectadas a ellos (dicha información puede ser también utilizada por PLC's, "Programmable Logic Controller". Generalmente este proceso se realiza para intervalos muy cortos. Será entonces cuando la MTU investigará los RTU's usualmente con una frecuencia menor, estos datos se chequearan para controlar que no se cumplan condiciones de alarma, y en caso de que una alarma se encontrase presente, esta sería catalogada y visualizada en las listas especiales de alarmas que la aplicación de SCADA posee.

Los datos procesados por SCADA pueden ser de tres tipos distintos:

- Datos analógicos que sean representados en gráficos (por ejemplo números reales).
- Datos digitales (on/off) que pueden tener o no alarmas asociadas a un estado o al otro.
- Datos de pulsos que normalmente se contabilizaran o se acumularan (por ejemplo las revoluciones de un medidor).

8.2.2. Prestaciones, requisitos y módulos de un SCADA

Un paquete de SCADA debe ofrecer las siguientes prestaciones:

- Posibilidad de creación de paneles de alarma que indiquen al operario la presencia de cualquier tipo de incidencia, y informe un registro de dichas alarmas.
- Ejecución de aplicaciones, que permitan la modificación, anulación o iniciación de las leyes de control y de las tareas del autómata, siempre bajo ciertas condiciones.
- Generación de históricos de la información recolectada en planta, que pueden ser volcados para su posterior uso en una hoja de cálculo.
- Posibilidad de programación numérica, que permita resolver cálculos aritméticos de elevada resolución sobre la CPU del ordenador.

Todas estas acciones se ejecutan mediante un paquete de funciones que permite su reprogramación al contener zonas de programación básica (como C, Pascal o Visual Basic), lo que aporta al sistema una grandísima versatilidad. Algunas aplicaciones SCADA incorporan unas librerías que permiten personalizar la aplicación en función de la necesidad de cada cliente.

Un sistema SCADA debe cumplir con diversos requisitos para que su instalación se vea aprovechada:

- Debe tratarse de un sistema de de arquitectura abierta, que sea capaz de adaptarse según las necesidades y los cambios de la empresa.
- Debe permitir una comunicación transparente entre el usuario y el equipo de planta, así como con el resto de la empresa (redes locales y de gestión).
- Debe consistir en una aplicación sencilla de instalar, que no ofrezca complicaciones en su ejecución y que tenga una interfaz amigable para el usuario.

Los módulos que incluye SCADA que son los encargados de gestionar las actividades de adquisición, control y supervisión de datos son los siguientes:

- Módulo de configuración: Es el módulo que permite modificar la interfaz de los programas en función de los requerimientos de cada cliente.
- Interfaz gráfico del operador: Este es el módulo encargado de que el usuario pueda monitorizar el control y la supervisión de los procesos que se llevan a cabo.
- Módulo de proceso: En función del valor de las variables del sistema, este módulo lanza acciones de mando preprogramadas.
- Gestión y archivo de datos: El módulo de gestión y archivo de datos se encarga de ordenar y almacenar los datos para que puedan ser procesados por cualquier otra aplicación.
- Módulo de comunicaciones: Se encarga de gestionar el intercambio de datos entre todos los niveles de arquitectura de SCADA, así como de realizar el intercambio de datos con el resto de herramientas informáticas de gestión.

8.2.3. Arquitecturas SCADA

Con el paso del tiempo los sistemas SCADA han ido evolucionando de la siguiente manera durante cuatro distinguidas generaciones:

- **Primera generación: "Monolítico"**

En sus inicios, los sistemas SCADA contaban con un sistema informático soportado en grandes minicomputadoras. Cuando se desarrollo SCADA, no existía ninguna red común y por ese motivo eran sistemas independiente sin conectividad con otros sistemas. Los protocolos de comunicación utilizados eran estrictamente protocolos de propietario para entonces. La seguridad del sistema estaba basada en una copia del sistema realizada en un ordenador central conectado a todos los RTU's, este sistema se utilizaba en caso de que hubiese algún error eventual del ordenador principal del sistema.

- **Segunda generación: "Distribuida"**

En esta segunda generación de SCADA, la información y el procesado de órdenes eran distribuidos a través de múltiples estaciones que estaban conectadas a través de una red LAN. La información se compartía prácticamente a tiempo real. Cada estación era responsable de una tarea en particular procurando que el tamaño y el coste de cada estación fuese menor que el utilizado en la Primera generación. Los protocolos de comunicación todavía no estaban estandarizados. Por ese motivo, al seguir utilizando un protocolo propietario, muy poca gente tenía el conocimiento necesario para determinar cuan segura era la instalación SCADA.

- **Tercera generación: " Conectada"**

De forma similar a una arquitectura distribuida, cualquier SCADA complejo puede ser reducido a su forma más simple, en cuanto a componentes se refiere, y conectada a través de protocolos de comunicación. En el caso del diseño conectado, el sistema puede ser esparcido a través de más de una red LAN y separado geográficamente. Diversas arquitecturas distribuidas de SCADA funcionando en paralelo, con un único supervisor y histórico de datos, podría ser considerado una arquitectura de red. Esta solución es mucho más efectiva en relación al coste en sistemas a gran escala.

- **Cuarta generación: "Internet de las cosas"**

Está arquitectura es una versión más comercial disponible de computación en la nube. Los sistemas SCADA han adoptado esta nueva tecnología de trabajar a través de internet para reducir de manera significante en la infraestructura de costes y incrementar la facilidad de mantenimiento e integración. Como resultado, los sistemas SCADA ahora pueden realizar informes de estado en tiempo real utilizando una escala horizontal disponible en la nube, de esta forma,

se pueden implementar algoritmos más complejos que son prácticamente imposibles de implementar en los tradicionales PLC's. Además, el uso de protocolos de conexión abiertos como son TLS("Transport Layer Security" o "Capa de Transporte de Seguridad"), provee al sistema de unos límites de seguridad más comprensibles y manejables que las mezclas heterogéneas de protocolos de redes propietario con diferentes implementaciones de SCADA descentralizados.

Parte III

MARCO APLICADO

9. Solución adoptada

9.1. Solución real/simulación

La solución que se adoptaría en un caso real sería la siguiente:

Primeramente, se realizaría la programación de los autómatas. Esta programación interna contaría tanto con el módulo de planificación de caminos, como el resto de la lógica propia de lo que debe hacer el robot en función de cada situación dentro del sistema logístico. No se realizará un estudio de dicha programación puesto que la longitud estimada del proyecto que se está realizando no lo permite.

Por otro lado, estos robots utilizarían un sistema de odometría basado en un complemento denominado *"roto-scan",* se puede observar un ejemplo de roto-scan en la figura 48.

Figura 48. Roto-Scan

Este dispositivo es un sensor óptico de distancia que también puede ser considerado una unidad de radar. Un rayo laser escanea una zona de trabajo semi-circular, si el rayo laser impacta contra una persona u objeto, la luz difusa que se refleja desde ese cuerpo es detectada por el receptor del dispositivo. En función de la señal recibida por el receptor, utilizando tanto el ángulo de radiación como el tiempo de propagación del sistema se calculan las coordenadas exactas del cuerpo que ha reflejado la luz.

En cuanto al principio de operación del roto-scan, este se basa en un diodo laser que produce un rayo de luz direccional sobre un espejo rotativo, este espejo cubre un área de 90º y mediante la rotación proporcionada por el motor adjunto a este se logra cubrir la totalidad del arco circular. Si el rayo que refleja el espejo impacta contra un objeto o persona dentro del área de trabajo, este es reflejado

y recogido nuevamente por el espejo que lo proyectará sobre el elemento receptor.

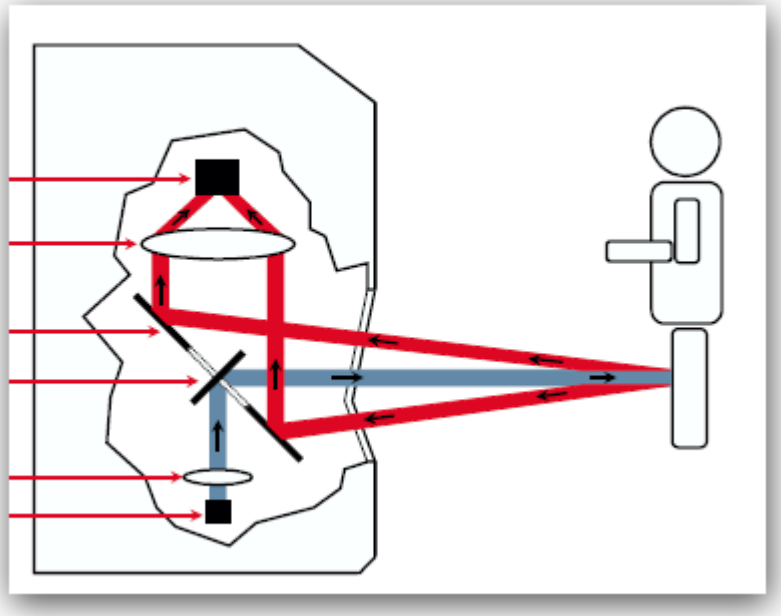

Figura 49. Método de operación del roto-scan

Pero desafortunadamente, no se cuenta ni con una nave industrial ni con robots para poder realizar el proyecto de forma real. Es por ese motivo que se ha decidido realizar una simulación del sistema mediante software.

La solución adoptada para el caso que se presenta mediante simulación es la siguiente:

Se ha generado una aplicación que permite al usuario realizar dos tipos de acciones distintas.

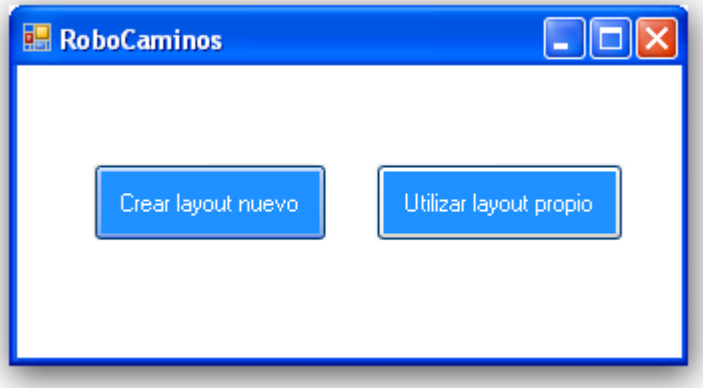

Figura 50. Aplicación creada para la realización de este proyecto

Empezando por el botón izquierdo de la aplicación, este redirige al usuario a un nuevo formulario en el que se le permite generar una distribución de planta de su nave industrial, formulario en el que debe indicar en qué posición se encuentran los nodos (almacenes y entradas y salidas de productos) más importantes. Se puede observar una captura de pantalla de dicho formulario en la figura 51.

Una vez realizada la distribución, se requiere que el usuario informe la distribución de caminos paso a paso entre cada uno de los nodos, debido a que no se cuenta con un sistema de odometría como el roto-scan. La aplicación permite realizar la panificación de caminos necesaria para que el robot se desplace desde un nodo inicial a un nodo final.

El próximo paso que se realizaría, sería llevar esta información a un siguiente nivel, en el cual se tendría que generar una imagen del sistema para que automáticamente se exportase a la aplicación SCADA y se simulase el sistema con dicha distribución en planta. Es una expansión del sistema que no se ha realizado y que se propone como propuesta de mejora, puesto que la extensión del proyecto obliga a acortar los ambiciosos objetivos que se podrían proponer.

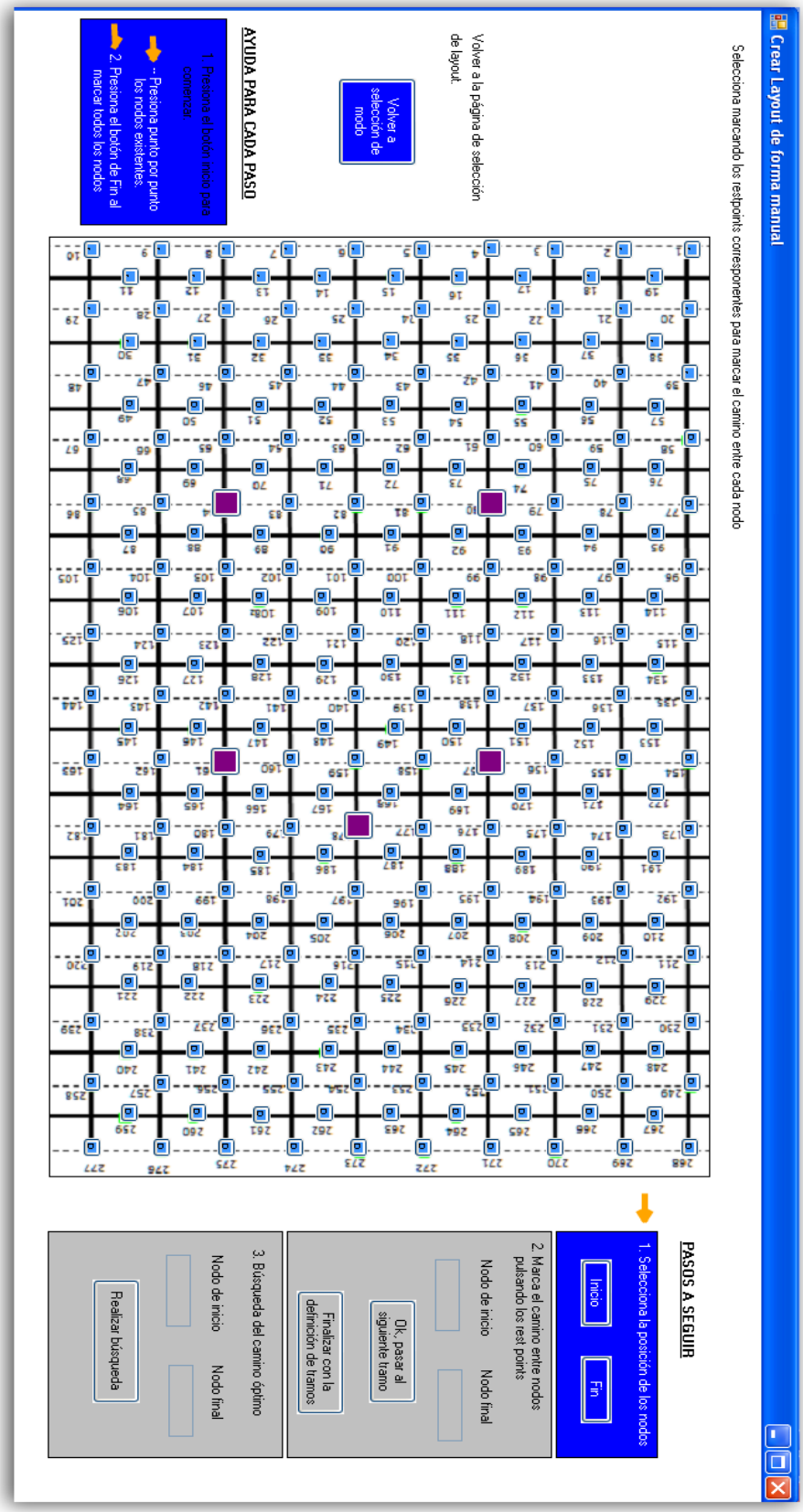

Figura 51. Formulario para la creación de nuevos layouts

Mirando ahora el segundo botón de la figura 50, al hacer click en dicho botón, este transporta al usuario al formulario que se muestra en la figura 52 mostrada a continuación.

Figura 52. Formulario para trabajar con el layout predeterminado

Este formulario es el activador de la simulación del transporte de hasta tres pedidos simultáneos para una distribución en planta predeterminada (se mostrará con detalle durante los próximos apartados de este proyecto). Una vez la orden se ha enviado, esta se podrá visualizar en la aplicación SCADA que se ha creado específicamente para esta distribución en planta, este tema será tratado más adelante con profundidad.

9.2. Arquitectura de software propuesta

Con la finalidad de solventar la problemática presentada, tal y como se ha ido mostrando en el apartado anterior, se ha desarrollado una arquitectura de software que permite simular el sistema de forma aproximada.

La primera aproximación realizada ha sido la creación y adaptación de una distribución en planta predeterminada o "*layout*", para poder llevar a cabo las simulaciones. El proceso llevado a cabo para decidir las características de dicho layout se muestran de forma detallada en el siguiente capítulo .

Con el layout caracterizado completamente se planteó el SCADA de iFix como método de representación y visualización del mismo, dicho SCADA presenta

importantes beneficios para obtener un sistema compacto. El funcionamiento y las características más importantes de la aplicación SCADA se discuten en el en proximos apartados de este proyecto. Se puede observar una captura de pantalla de la visualización de SCADA en la figura 53.

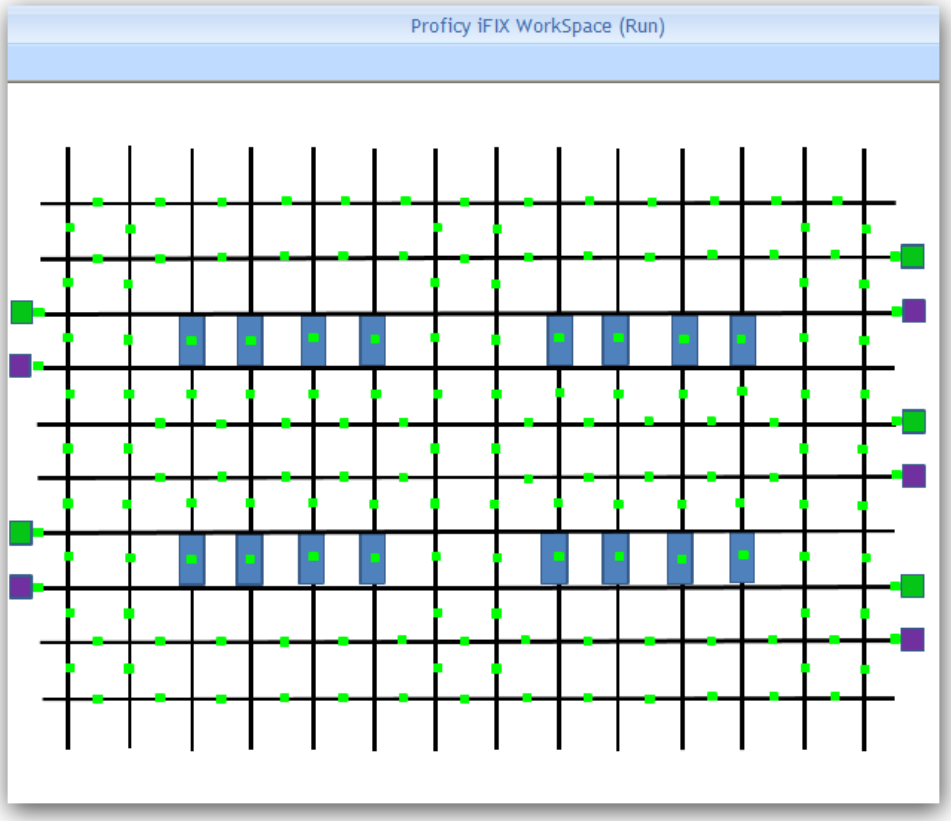

Figura 53. Figura de SCADA en su funcionamiento sin robots en marcha.

Aprovechando que SCADA iFix posee buena sinergia con muchos, pudiendo perfectamente decir cualquiera, servidores y clientes OPC se ha tomado la decisión de utilizar el servidor OPC de KEPWARE 'KEPserverEx 5'. Dicho servidor OPC, mediante el driver Advanced Simulator que incorpora, simula el movimiento de la flota de robots al contener un tiempo de refresco regulable para cualquiera de los 'tags' que se le añadan. Se realizará una explicación detallada de este sistema en próximos capítulos del proyecto. Se adjunta una captura de pantalla de dicho servidor OPC a continuación mostrando la disposición de los robots.

Figura 54. Flota de robots dentro del OPC Server

El driver Advanced Simulator que se menciona en el párrafo anterior precisa de una base de datos sobre mediante la cual poder actualizar el valor de los 'tags' del servidor OPC, y de dicha manera simular el movimiento de los robots. Una de las opciones que dicho driver permite utilizar son las columnas de una tabla SQL. De esta manera, se ha generado una tabla por robot para guardar las posiciones que estos irán siguiendo. En próximos capítulos se realizará un estudio más detallado de todo este sistema. A continuación se muestra una captura de pantalla de la disposición de las tablas generadas para la resolución de este proyecto.

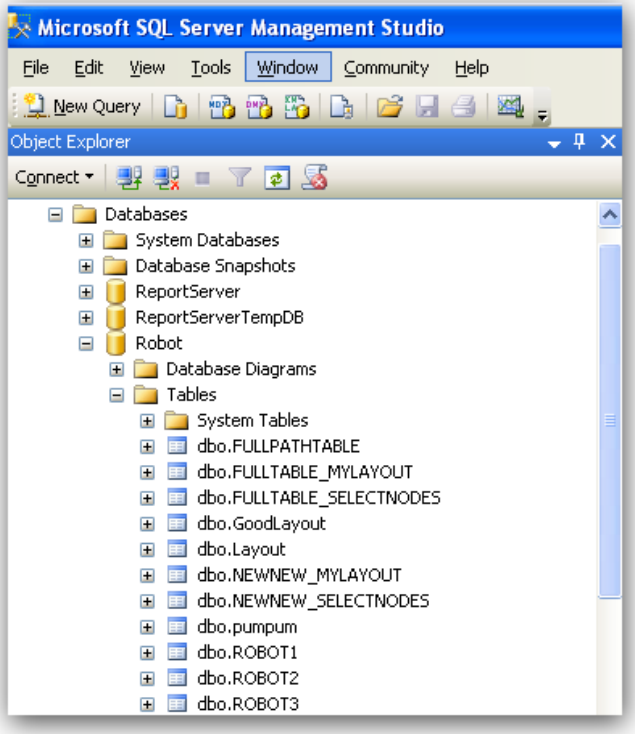

Figura 55. Distribución de algunas de las tablas creadas para este proyecto

Para rellenar las posiciones que los robots irán adquiriendo en dichas tablas se ha generado una aplicación en Visual Basic a la que se ha llamado "**RoboCamino"** que las rellena siguiendo una lógica que será explicada en próximos apartados de este proyecto.

Una vez se ha explicado toda la jerarquía de programación, en la siguiente figura, se muestra la arquitectura de software en diagrama de flujo que compone la aplicación global.

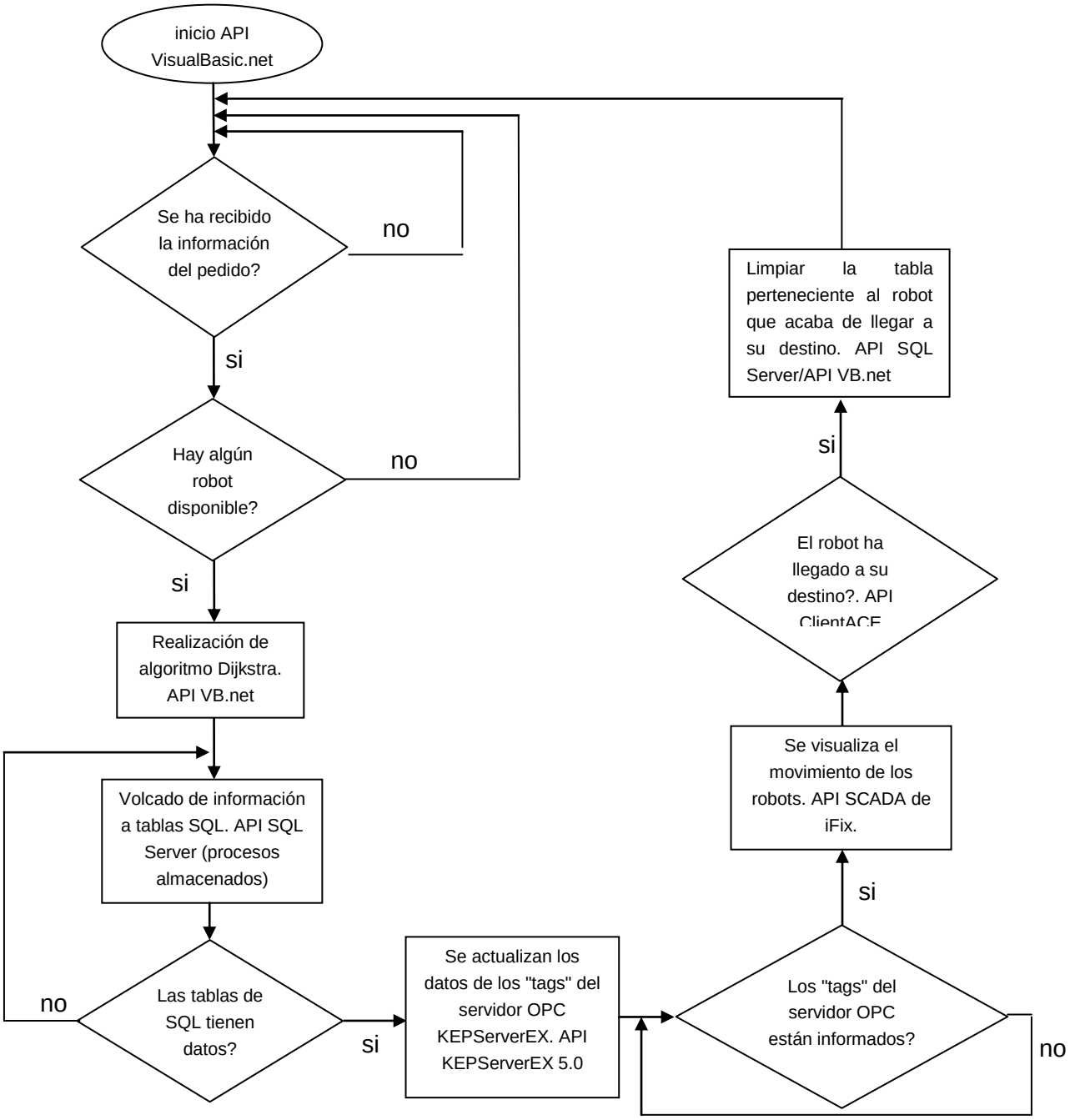

Figura 56. Diagrama de flujo del programa global

10. Definición del problema a resolver

En este apartado se tratarán los pasos que se siguieron para conseguir la estructura en planta o *"layout"* definitivo.

10.1. Descripción del layout

La primera aproximación que se realizó para el sistema fue la que puede observarse en la figura 57.

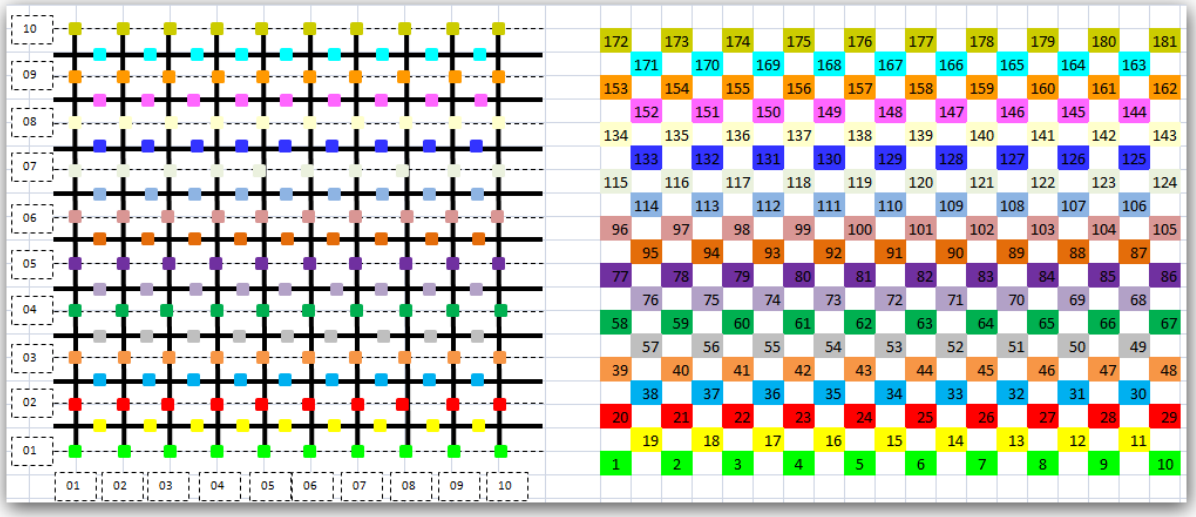

Figura 57. Aproximación inicial

181 nodos, distribuidos entre 10 filas y 10 columnas iguales. Dada la suposición que puede observarse en la siguiente figura, se disponía de una nave industrial de 36 m^2:

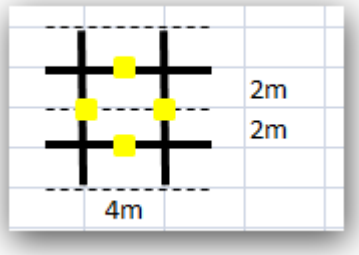

Figura 58. Medidas de la nave industrial

A sabiendas de que se disponía de dicha nave industrial se realizó el primer planteamiento de disposición interna:

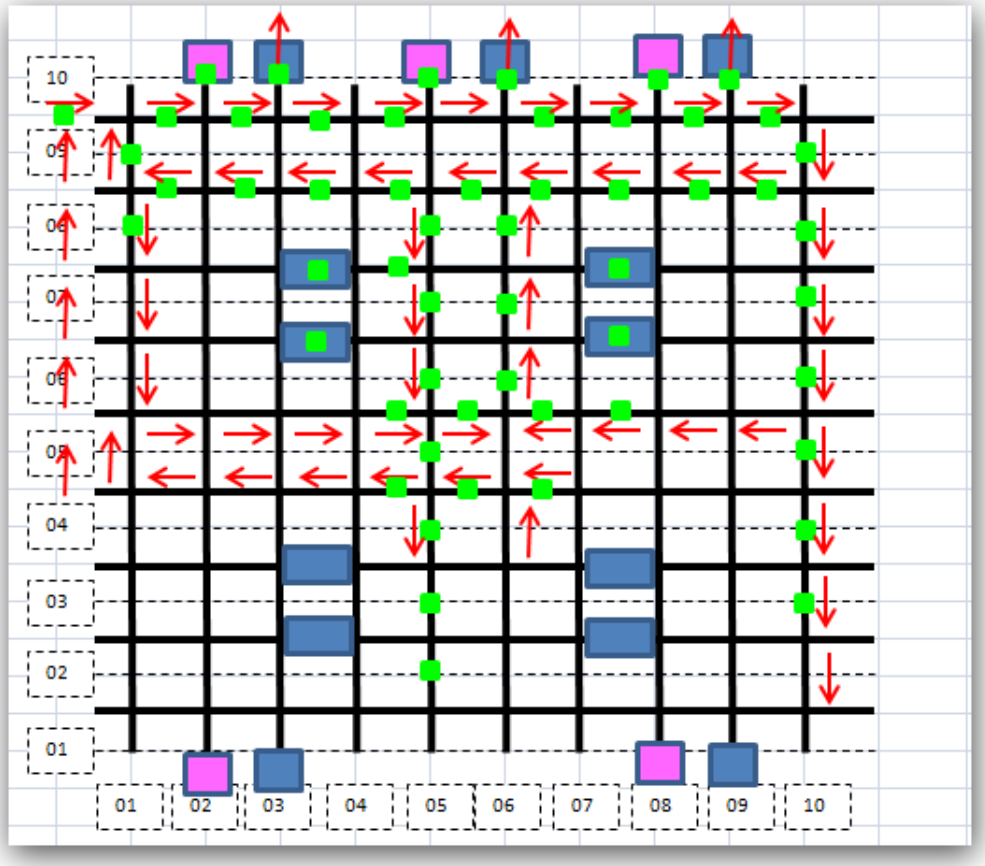

Figura 59. Primer planteamiento de disposición interna

En este planteamiento de disposición interna se distinguieron tres tipos diferentes de objetos dentro de la nave industrial:

Tren de Salida

Tren de entrada

Almacén

Como se puede observar, se ha tratado de realizar un lazo en dos direcciones a cada lado de los almacenes para un más fácil acceso a cualquiera de los almacenes desde cualquier punto de la nave industrial.

Pero en la parte baja de la nave no se ha logrado dicha disposición por falta de espacio. Por ese motivo, se decidió incrementar la longitud de la nave de forma que se pudiesen implementar dichos lazos por los cuatro costados de cada grupo de almacenes. Ya puestos, también se decidió incrementar el número de almacenes. Quedando la siguiente estructura definitiva:

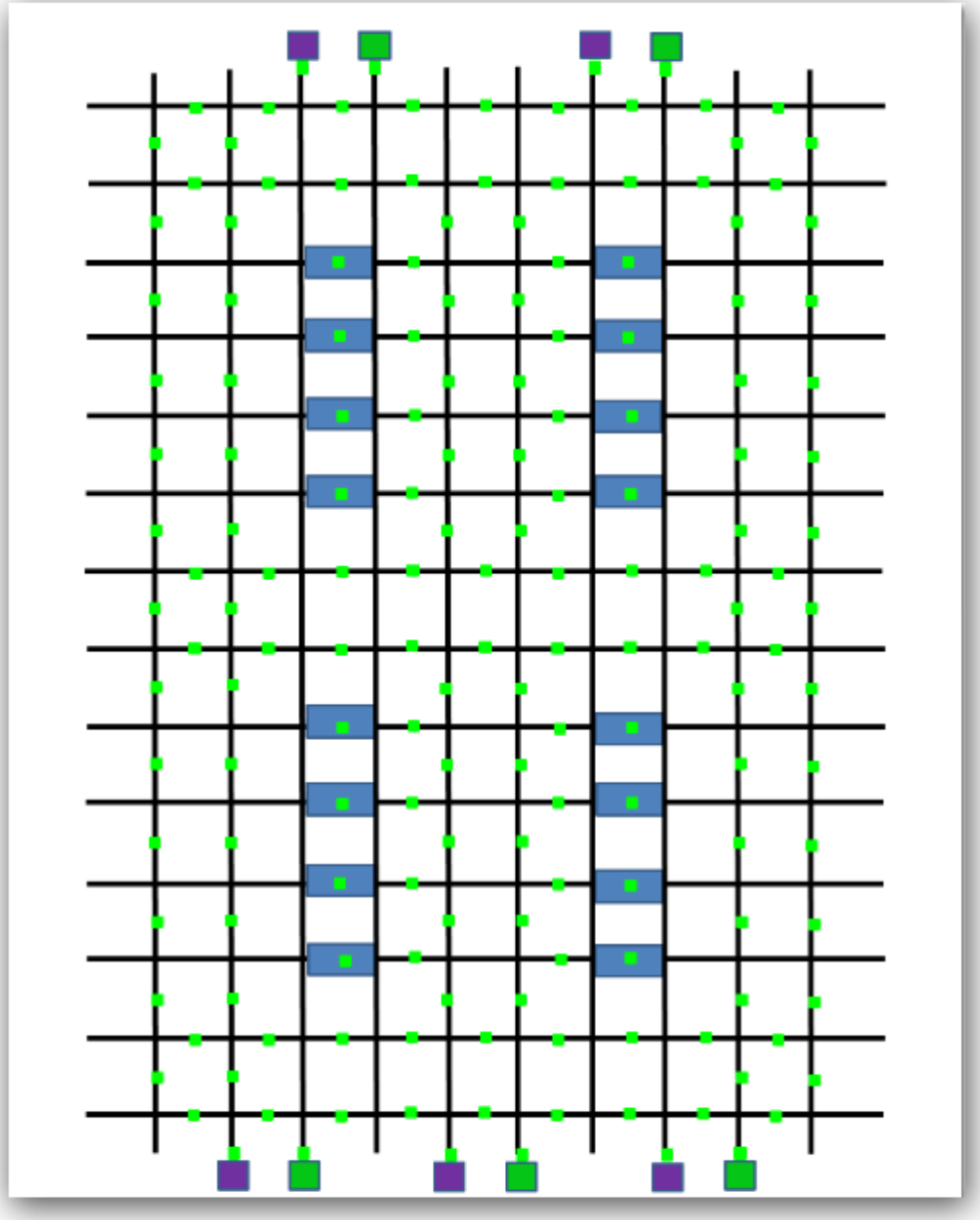

Figura 60. Estructura defintiva en planta del "layout"

Como puede apreciarse, no han variado los elementos internos del layout respecto de las primeras aproximaciones. Pero ahora se ha logrado el espacio necesario para poder realizar los lazos alrededor de todos los almacenes tal y como se muestra en la figura 61.

10.2. Simplificaciones impuestas

Por motivos de sencillez, debido a que la longitud teórica del proyecto no lo permitía, se han impuesto una serie de simplificaciones que serán nombradas a continuación:

- Los robots no podrán desplazarse libremente por la nave, los caminos tienen direcciones, y los robots estarán obligados a seguirlas siempre siguiendo las directrices marcadas con anterioridad. Puede observarse este hecho en la figura 61.
- Se limitará el número de robots simultáneos en funcionamiento a 3. Debido a que el incremento de robots incrementa la complejidad del sistema.

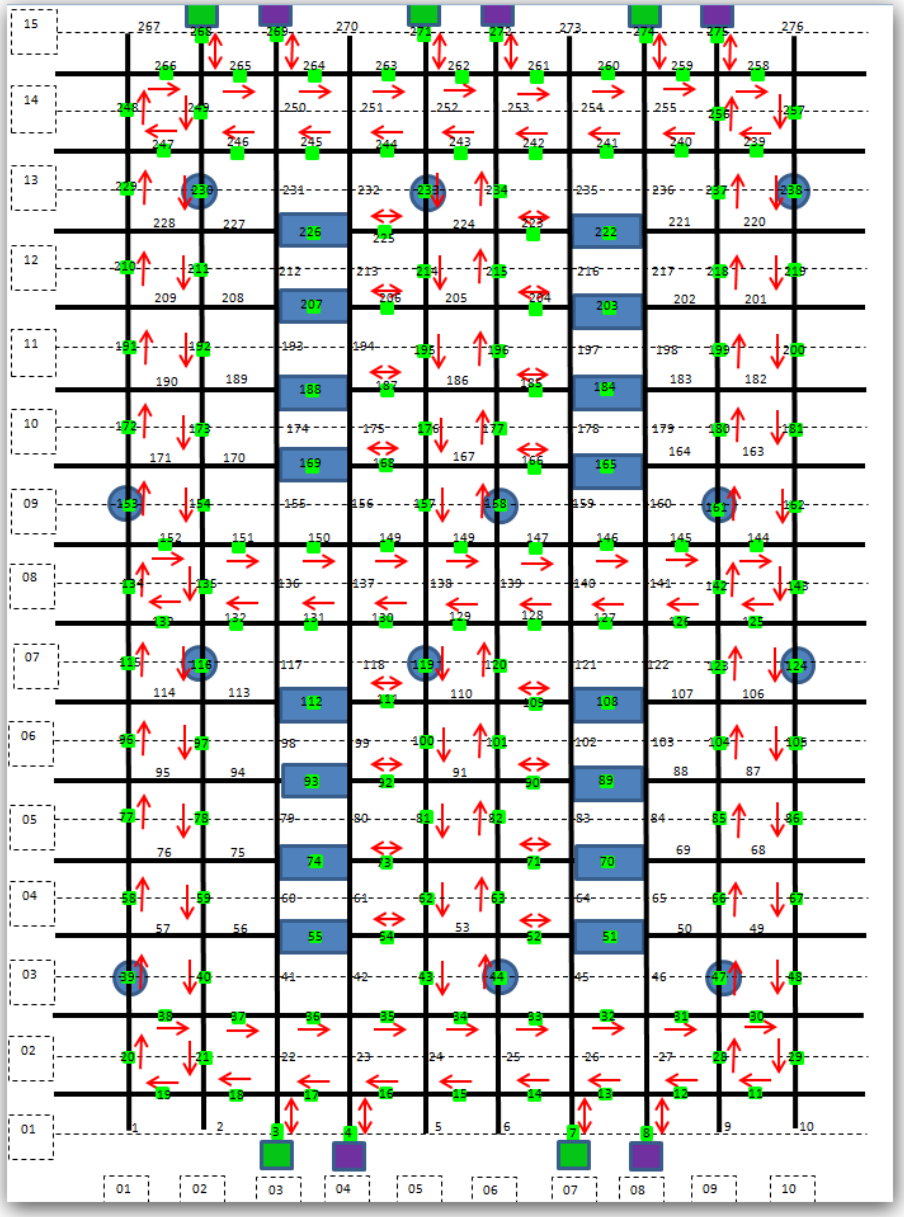

Figura 61. Layout definitivo con direcciones implementadas

Visualizando la figura anterior, puede observarse que se ha añadido un nuevo elemento con respecto al planteamiento que se hacía inicialmente, ahora el sistema cuenta con las siguientes particularidades:

 Zona de almacenaje.

 Tren de entrada de producto.

 Tren de salida de producto.

 Nodos virtuales.

Es sencillo entender porque han aparecido en el sistema estos nuevos elementos que han sido nombrados nodos virtuales. El modelo previo, el cual era exactamente el mismo al actual pero eliminando los nodos virtuales, ya se pensaba definitivo. Por ese motivo, se realizó el grafo del algoritmo Dijkstra de forma manual para poder comprobar las dimensiones que dicho grafo tendría. Fue entonces cuando se observó que realizar una búsqueda de caminos en ese sistema no tenía sentido, puesto que el camino entre dos posiciones no tenía ninguna variante. Es decir, era un grafo lleno de caminos directo entre inicio y fin del trayecto.

Entonces se decidió añadir unos nodos intermedios (inexistentes) que añadiesen una variedad de opciones en cuanto al algoritmo de Dijkstra representa. Así aparecieron los llamados nodos virtuales. Puede observarse el grafo Dijkstra definitivo en el Anexo 1.

11. Explicación detallada de las aplicaciones que forman la solución propuesta.

Este apartado es el corazón del marco aplicado. Aquí se detallan con precisión todas y cada una de las particularidades que contienen las aplicaciones que dan forma a este proyecto.

Se ha distribuido este capítulo en función de las cuatro aplicaciones básicas que conforman la creación de este proyecto. Por orden de aparición en el capítulo, aplicación de visual basic.net (incluyendo el algoritmo Dijkstra), base de datos SQL Server, archivo de servicio OPC y aplicación SCADA.

11.1. Aplicación de Visual Basic .net

11.1.1. Algoritmo Dijkstra

Para realizar el algoritmo de Dijkstra se ha aprovechado un código creado por una persona anónima. Dicho código ha sido modificado en función a la conveniencia de este proyecto. No se ha introducido el código realizado en Anexo puesto que la magnitud del mismo era demasiado elevada.

Es necesario añadir que se han realizado dos versiones distintas del algoritmo Dijkstra, en función de las necesidades de cada formulario dentro de la aplicación.

Para acceder al primer punto en el que se realiza la búsqueda de caminos, hay que realizar los siguientes pasos:

- Primeramente, hay que acceder al formulario de creación de un nuevo layout como se muestra en la figura 62.

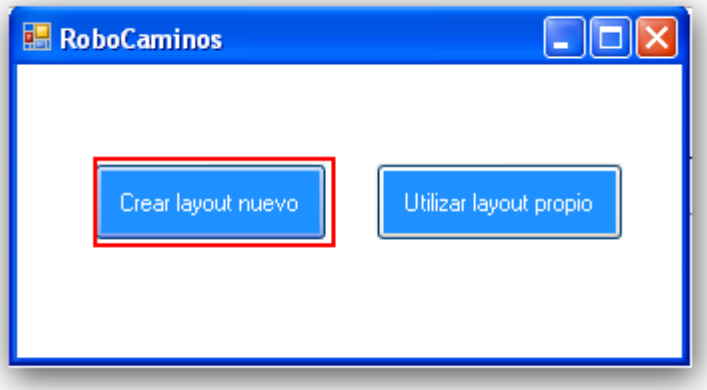

Figura 62. Aplicación RoboCaminos

- En segundo lugar, una vez ya entrado en dicha parte de la aplicación se ha de realizar todos los pasos indicados para la creación del mismo (se detalla cómo realizar dichos pasos en los próximos apartados de este capítulo).
- Una vez todos los pasos para la creación de la nueva distribución en planta hayan sido realizados, al pulsar el botón que se muestra en la figura 63, se realiza el algoritmos de dijkstra para la distribución seleccionada con anterioridad, entre los dos nodos seleccionados mediante los cuadros de dialogo que se pueden observar en la siguiente figura.

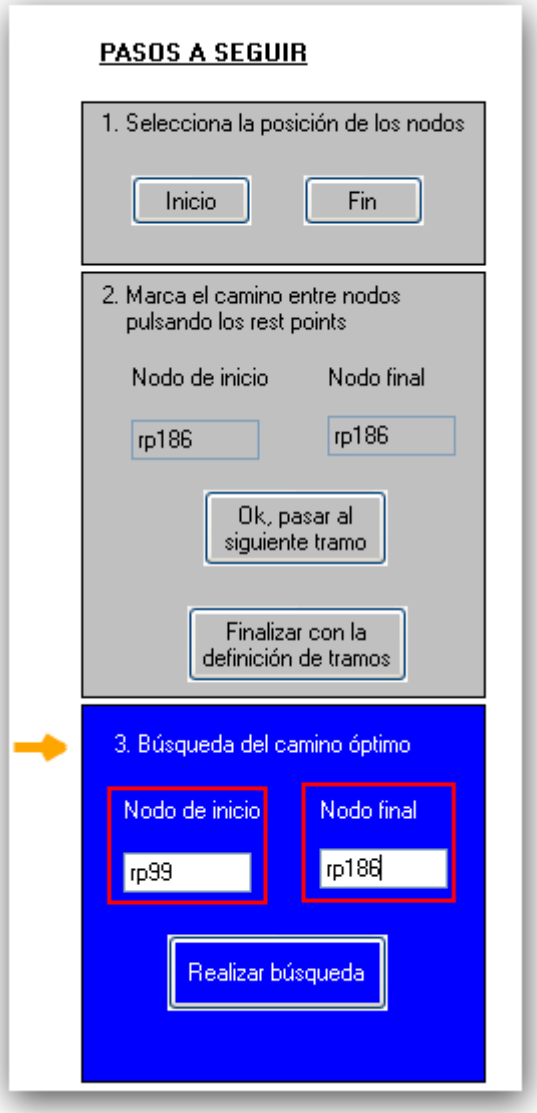

Figura 63. Realización de búsqueda de camino óptimo

- Una vez pulsado el botón "Realizar búsqueda", se ejecutará la parte de código que realiza la búsqueda del camino óptimo y se mostrará por pantalla la distribución de nodos que seguirá el robot, tal y como se muestra en la figura 64.

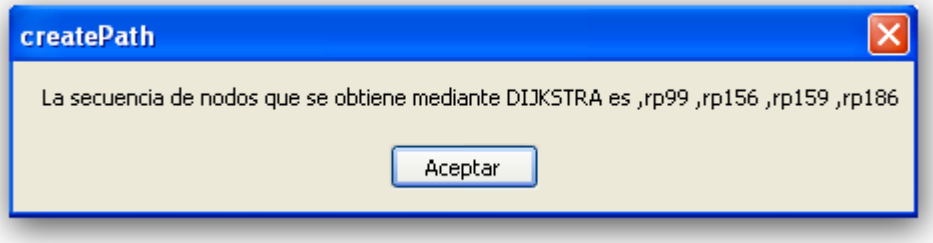

Figura 64. Secuencia de nodos que realizará el robot

Por otro lado, existe un segundo punto, menos visual que el anterior pero con más dificultad técnica a la hora de llevarlo a cabo, en el cual el programa realiza el algoritmo de búsqueda Dijkstra. No se procederá con la explicación exhaustiva de lo que sucede una vez se realiza la búsqueda de caminos de esta manera, puesto que se abordará desde otros apartados dentro de este mismo capítulo.

Para acceder a dicho punto se deben seguir los siguientes pasos dentro de la aplicación:

- En primer lugar, se debe acceder al formulario del aplicativo en el cual se aprovecha la distribución en planta predeterminada que se mostraba en el apartado anterior. Para ello se pulsará al botón que se muestra seleccionado en la figura 65.

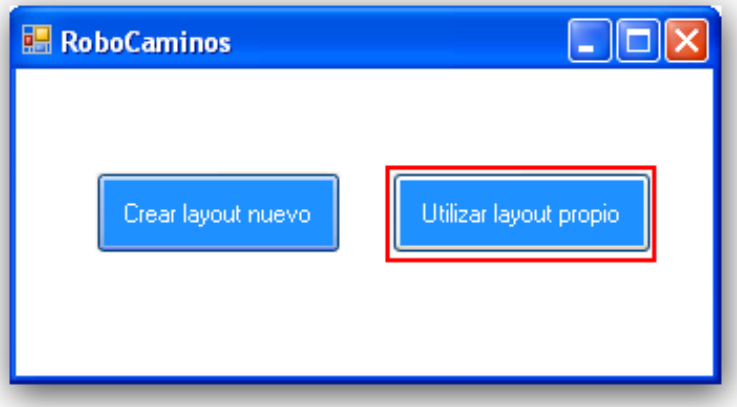

Figura 65. Acceso a la segunda aplicación del algoritmo Dijkstra

- Posteriormente, dentro del formulario que puede observarse en la figura 65, se debe introducir los nodos de inicio y de fin entre los cuales el robot debe desplazar el producto.

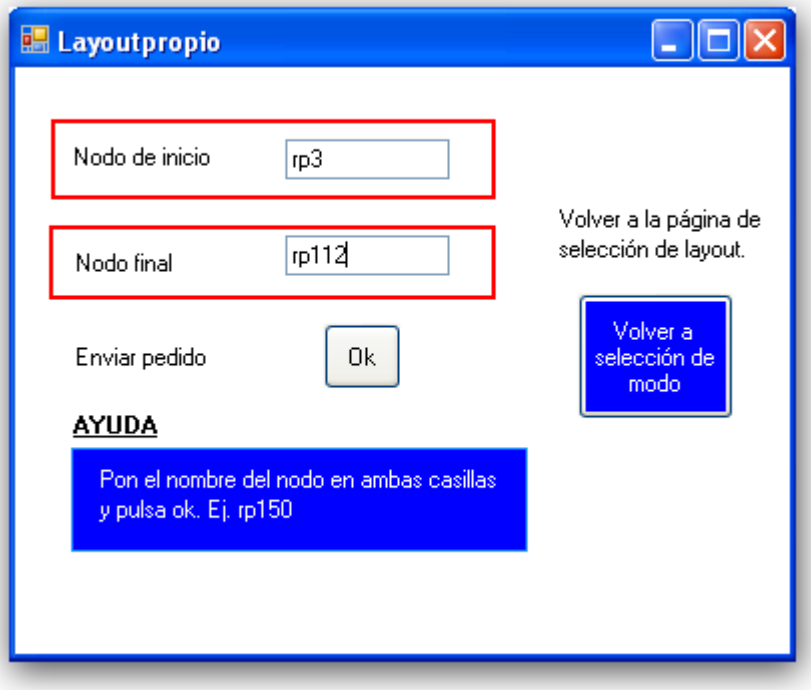

Figura 66. Segundo punto de aplicación del algoritmo Dijkstra

- Al enviar el pedido pulsando "Ok" se realiza el algoritmo de búsqueda, con la particularidad de que esta vez no se muestra por pantalla. Sin embargo, esta vez se rellena una tabla que almacena todas y cada una de las posiciones que el robot tendrá mientras se encuentre entre el nodo inicial y el nodo final. Se puede observar más detalladamente este proceso en los próximos apartados de este mismo capítulo.

El algoritmo de Dijkstra siempre se ejecuta sobre un grafo, en este caso el grafo es conocido, puesto que se está trabajando en base a una distribución en planta ya predeterminada. Se ha realizado un esbozo de dicho grafo para tener una idea aproximada de las dimensiones del problema que se está trabajando, puede observarse el grafo en base al cual el algoritmo funciona en el Anexo 1.

11.1.2. Funcionamiento de la aplicación

La aplicación de visual basic tiene dos funcionalidades distintas que pasan a ser expuestas a continuación.

- **Formulario de creación de un nuevo "*layout*".**

Para acceder al formulario de creación de un nuevo "*layout*" es necesario pulsar el botón izquierdo de la pantalla de selección, tal y como puede observarse en la figura 61 expuesta con anterioridad.

Una vez dentro del formulario, la primera visión que tendremos del programa será la que puede observarse en la figura 69.

Mirando dicha figura que ha sido nombrada con anterioridad, se pueden diferenciar tres zonas distintas. Cada una de ellas ha sido marcada de un color distinto para una diferenciación más visual.

La primera zona a comentar, ha sido la marcada con color amarillo, puede observarse una visión de dicha zona separada del resto del formulario en la captura que se muestra a continuación.

Figura 67. Botón para volver a la selección de modo

Este botón fue implementado para el caso de querer cambiar de modo una vez iniciado un formulario. Al pulsar dicho botón, te devuelve a la pantalla incial para poder seleccionar nuevamente el tipo de layout, predeterminado o nuevo. Una vez apretado dicho botón, también se refrescan todas las variables iniciadas en el modo anterior, para una posible nueva utilización de las mismas en caso de seleccionar nuevamente el formulario para la creación de un nuevo layout.

La segunda parte a comentar y tener en cuenta es la parte marcada en color azul cian en la figura 69, que puede observarse aislada en siguiente imagen.

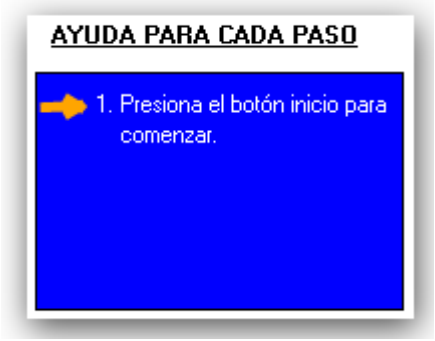

Figura 68. Ayuda para cada paso

En la ayuda paso para cada paso que ha sido mostrada en la figura 68, se informa una pequeña ayuda de cara al usuario para facilitar la realización del programa y hacer una interfaz más amigable para el mismo.

Esta ayuda irá siendo modificada según el avance del programa, justamente después de presionar el botón para abrir el formulario de creación de "*layout*" aparecerá la información que se ha mostrado en las anteriores capturas de pantalla, "Presiona el botón de inicio para comenzar".

Por otro lado, otro de los elementos a destacar dentro del entorno de programa es la parte central o nueva distribución en planta. Existen 276 posiciones sobre las cuales añadir nodos. Donde por nodos, se sobreentienden esas posiciones que son más importantes dentro de la nave industrial, tales como almacenes, puntos de entrada y salida o los nodos virtuales que han sido explicados en el punto anterior.

Las 276 posiciones sobre las cuales se puede añadir un nodo son pulsadores con distintas finalidades que serán detalladas más adelante.

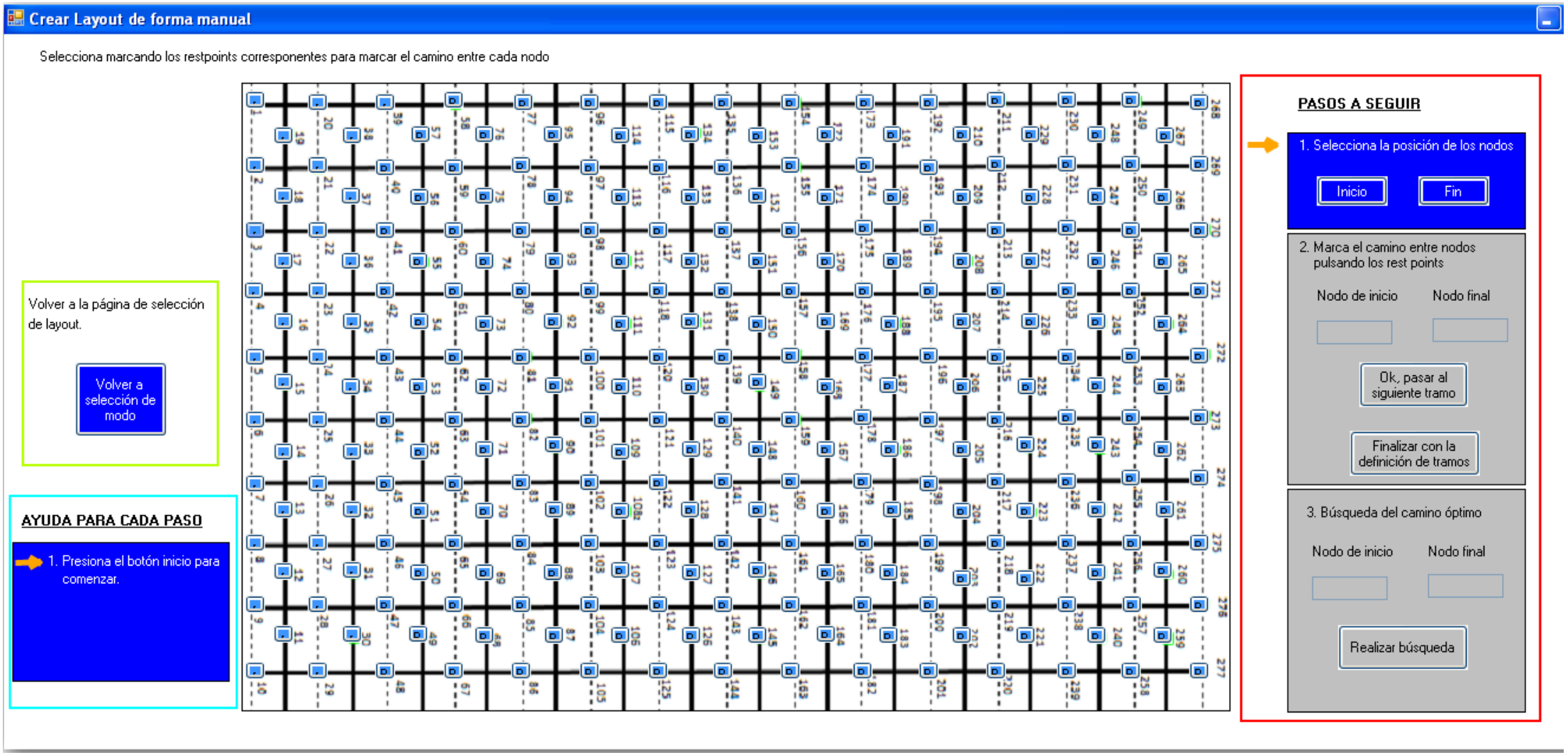

Figura 69. Formulario de creación de un nuevo "layout"

En último lugar cabe destacar la zona marcada con color rojo en la figura 69, que también puede observarse aislada en la imagen que se muestra a continuación. En función de esta captura se procederá al desarrollo de este apartado.

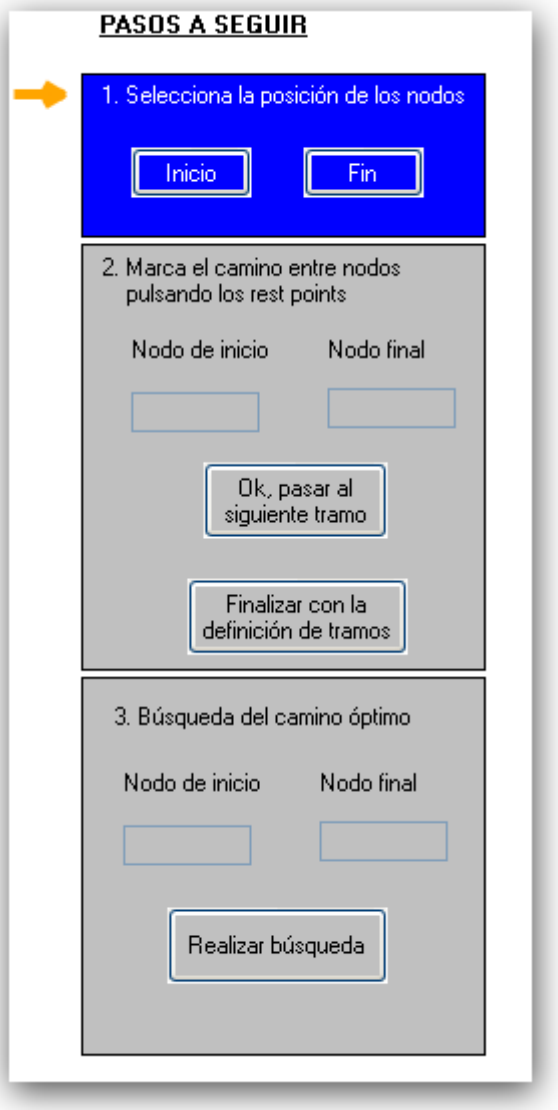

Figura 70. Guía interactiva de pasos a seguir por el usuario

Esta guía paso a paso cuenta con la particularidad de marcar el punto actual en el que se encuentra la simulación. Esta señalización se efectúa de maneras diferentes, la primera mediante una pequeña flecha que señala el bloque en el que se encuentra el programa, y la segunda consiste en marcar de color azul el bloque de trabajo en el cual se encuentra la simulación.

Una vez realizada una descripción global del entorno de trabajo se procederá con el desarrollo de la aplicación mediante un ejemplo sencillo.

Para empezar, hay que fijarse en las instrucciones que el programa ofrece desde dos fuentes distintas, la ayuda al usuario y la guía de pasos a realizar.

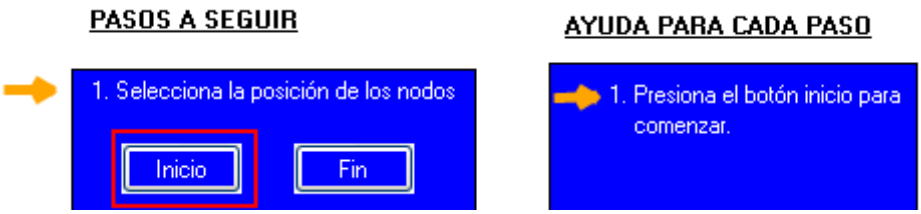

Figura 71. Primer bloque de pasos a seguir

Al pulsar el botón marcado en la figura anterior, el sistema lanzará el siguiente pop-up:

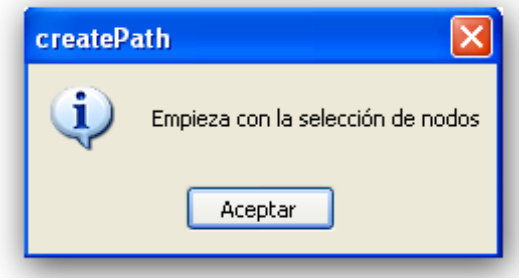

Figura 72. Inicio de selección de nodos

Tal y como puede observarse durante la simulación del programa, la "Ayuda para cada paso" ahora se ha visto modificada.

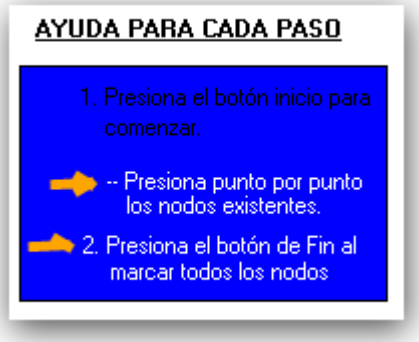

Figura 73. Ayuda para cada paso modificada

La ayuda informa que hay que ahora es el momento de pulsar uno por uno todos los nodos existentes, tal y como se ha comentado con anterioridad, se entiende por nodo cualquier punto importante del sistema (almacenes, entradas o salidas de producto, nodos virtuales, etc.). Puede visualizarse en la figura 74 un ejemplo después de haber pulsado sobre todos los nodos.

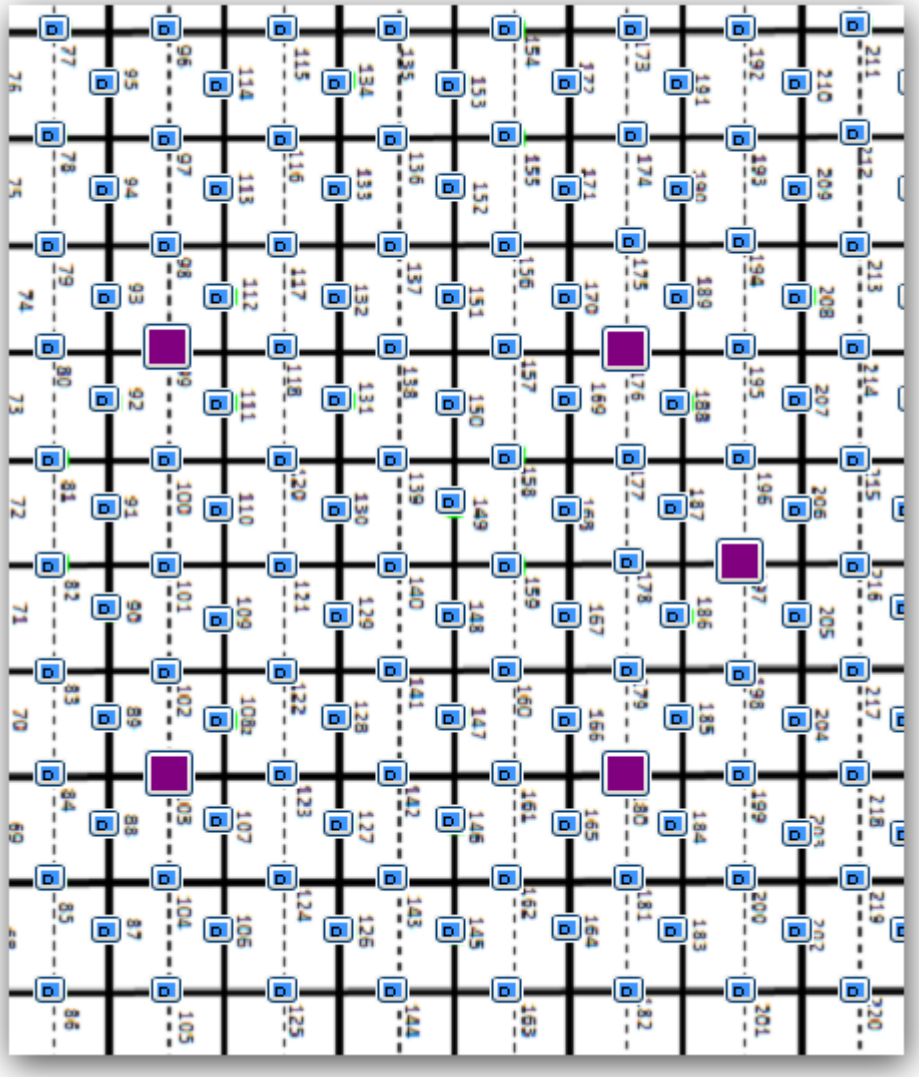

Figura 74. Layout con los nodos seleccionados

A continuación, al pulsar el botón "Fin" de la guía tal y como indica la ayuda para el usuario, se producen diversos cambios en el entorno de trabajo. El primero de ellos es que la guía de pasos a seguir ha cambiado de bloque tal y como se puede ver en la figura 75. Dentro de este bloque, se puede observar que automáticamente han aparecido los números de los "rp" (rest points o puntos de descanso), correspondientes a dos de los nodos seleccionados, dentro de los cuadros de diálogo. Por otro lado, el objeto de texto "Nodo inicial" ha sido resaltado en color amarillo mientras que el objeto de texto "Nodo final" lo ha sido en color naranja. Esto es para que el usuario relacione este hecho con lo que está sucediendo en la parte central de la pantalla, tal y como podrá observarse en la figura 76.

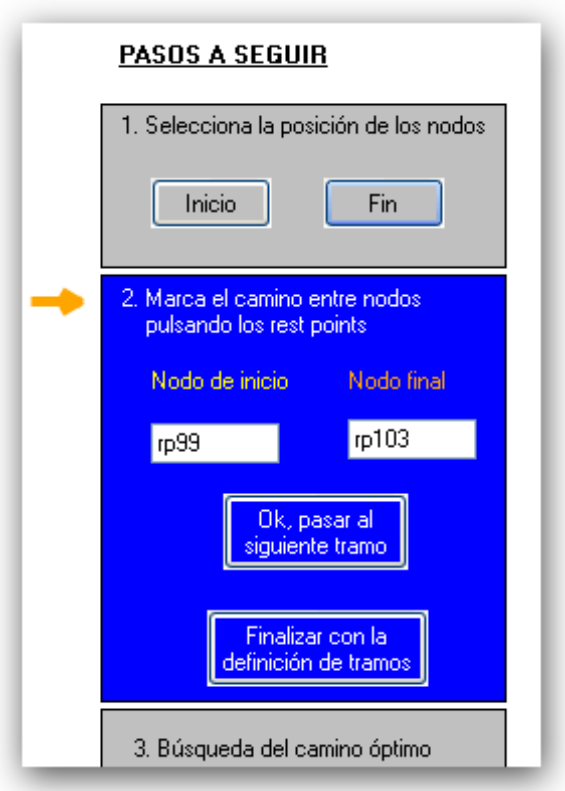

Figura 75. Activación del segundo bloque de pasos a seguir

Hablando ahora de la casuística que se muestra en la figura 76, se pregunta al usuario si existe camino entre los nodos que han sido marcados por la aplicación. Tal y como se ha hecho mención anteriormente, se le da el color amarillo al nodo inicial y el color naranja al nodo final.

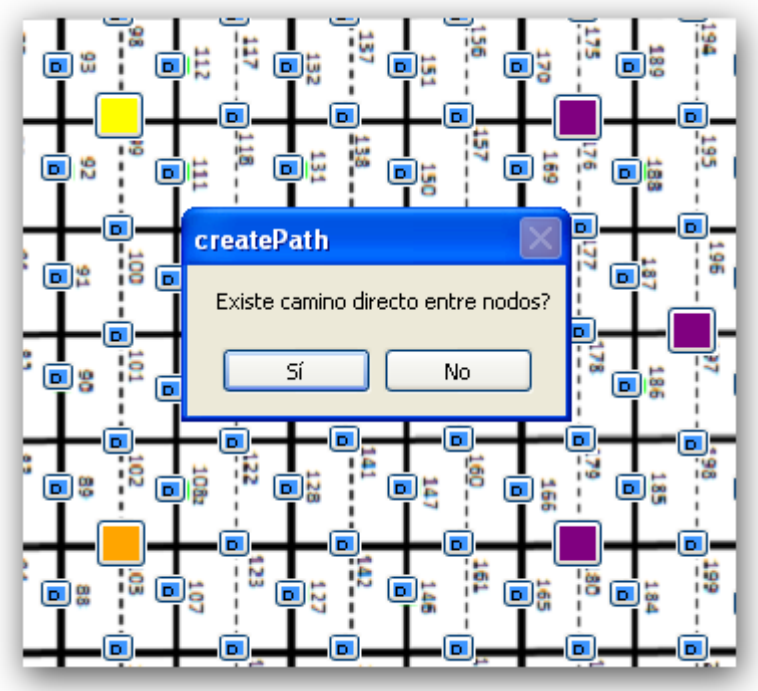

Figura 76. Pregunta sobre la existencia de camino entre nodos

Otro de los cambios del entorno de la aplicación ha acontecido en la ayuda para cada paso para el usuario. Se han modificado todas y cada una de las instrucciones, tal y como puede comprobarse en la figura 77.

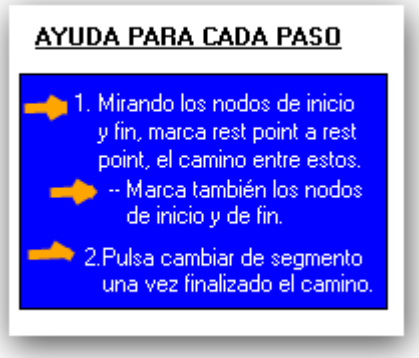

Figura 77. Ayuda para cada paso modificada (2)

Finalmente, la última modificación que puede apreciarse en el formulario consiste en un botón para retroceder el último rest point que se haya marcado. De esta manera en caso de que el usuario se equivoque siempre puede retroceder el último paso dado. Se puede observar dicho botón en la figura 78.

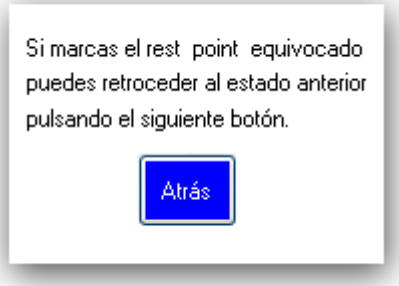

Figura 78. Retroceso del último rest point marcado

Una vez chequeados todos los cambios que ha sufrido el sistema, es el momento de continuar con los siguientes pasos.

Ahora aparecen dos posibles opciones ante el usuario, dar una respuesta positiva en caso de que sea posible el acceso directo entre nodos o dar una negativa. En caso de dar una respuesta positiva el usuario tendrá que marcar el camino que tendrá que hacer el robot entre un nodo y otro. Tal y como se muestra en la figura 79.

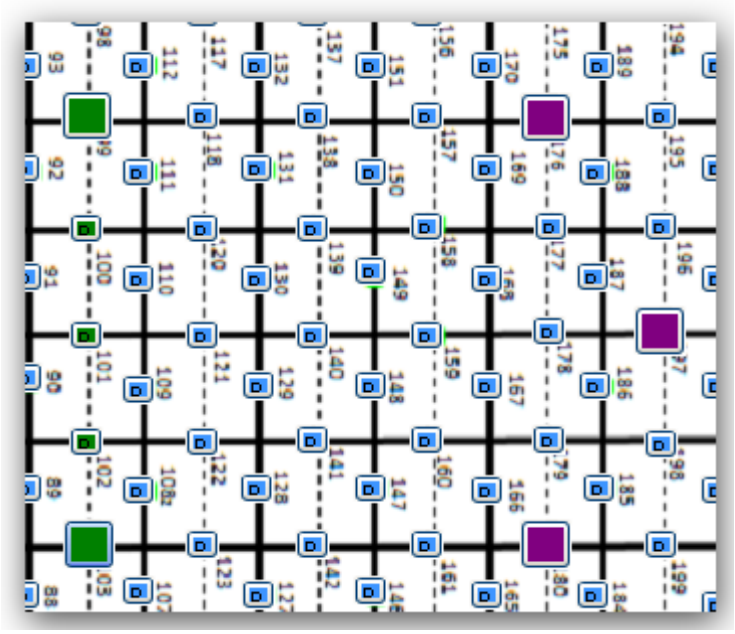

Figura 79. Marcar camino entre nodos

Con la finalidad de emular la función que un sistema de odometría como el comentado en la unidad 9 está funcionando en el sistema, se realiza el marcado de "rest point" entre nodo y nodo simulando todos los pasos que el robot dará desde inicio hasta fin. Al ir pulsando cada "rest point", tal y como la ayuda para el usuario indica, estos irán modificando su color para mostrar al usuario qué punto ha marcado ya. Una vez haya sido marcada toda la secuencia, pulsaremos el botón de la guía para cambiar de tramo, tal y como indica la ayuda. El programa recalculará los nodos de inicio y de final y volverá nuevamente a reformular la pregunta anterior hasta haber iterado sobre todas las combinaciones de nodos.

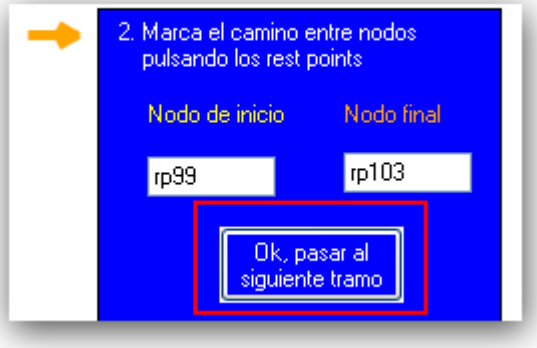

Figura 80. Paso al siguiente tramo

Por otro lado, en caso de dar una respuesta negativa a la pregunta que se muestra en la captura de pantalla de la figura 76, el sistema recalculará nuevamente el nodo inicial y final hasta recibir una respuesta afirmativa. En caso

de no recibir respuesta afirmativa, finalmente el sistema acabaría pasando al tercer bloque.

El paso siguiente consiste en finalizar con toda la odometría correspondiente al layout definido. Para el ejemplo que se está desarrollando se utilizará el mismo grafo que se mostró para realizar la explicación del algoritmo Dijkstra, que es el que puede observarse a continuación

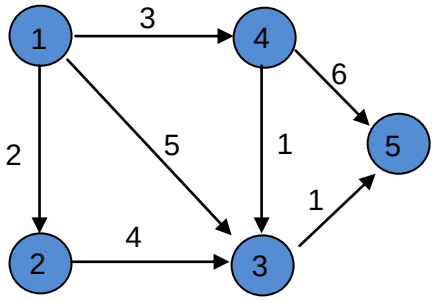

Figura 81. Grafo para el desarrollo del ejemplo Dijkstra

Una vez la odometría ha sido señalada, es decir, todas las posibles combinaciones de nodos han sido revisadas, aparecerá el siguiente mensaje por pantalla:

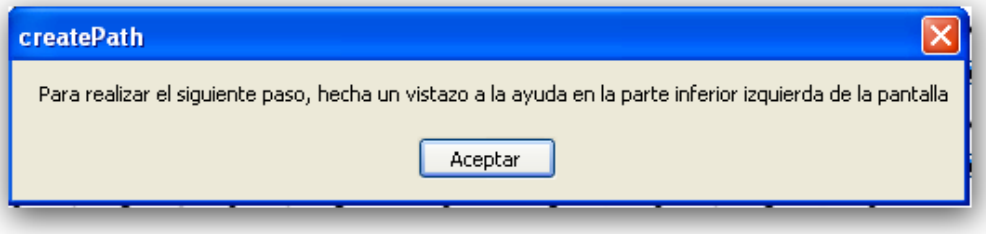

Figura 82. Pop-up de inicio del tercer bloque

La figura 81 indica el cambio al tercer y último bloque de esta parte de la aplicación. En esta última fase también pueden observarse diferentes cambios en la interfaz del programa.

El primero de los cambios que pueden apreciarse consiste en el paso al tercer bloque dentro de la guía de pasos a seguir. Puede observarse dicha modificación en la figura 83 que se muestra a continuación.

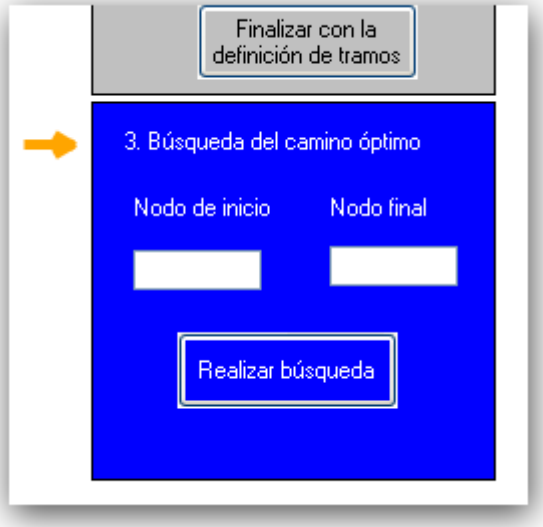

Figura 83. Cambio al tercer bloque de la guía paso a paso

En este nuevo bloque, se realiza el algoritmo Dijkstra tal y como se ha mostrado en anteriores puntos de esta misma unidad. Únicamente hay que poner los nombres del nodo de inicio y del nodo final y pulsar el botón "Realizar búsqueda".

Para realizar bien este último paso se ha realizado una última modificación a la interfaz, se trata de los mensajes de ayuda para el usuario. Los mensajes para el último tramo se muestran en la figura 84.

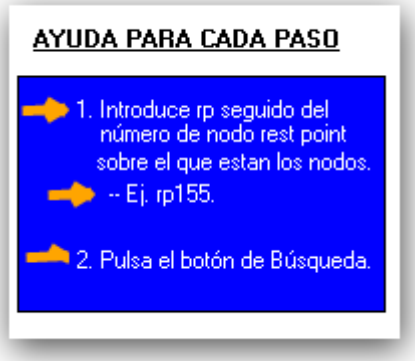

Figura 84. Ayuda para cada paso modificada (3)

Una vez chequeados los nuevos cambios sufridos en el sistema, es el momento de finalizar el ejemplo demostrando que el algoritmo Dijkstra se efectúa correctamente. Mirando el grafo de la figura 81, se puede hacer la siguiente correspondencia entre los nodos del grafo y los "rest points" de la simulación que se está llevando a cabo:

Tabla 1. Correspondencia entre nodos y rp

Nodo en el grafo	RP en la simulación
1	rp99
2	rp103
3	rp179
4	rp175
5	rp196

Entonces en el supuesto de que quisiera realizarse el algoritmo dijkstra entre los nodos 1(rp99) y 5(rp196), el camino óptimo si se observa el grafo sería 1-4-3-5. Por tanto, la solución que el programa debería mostrar sería rp99-rp175-rp179-rp196. Se puede observar el resultado de la simulación en la próxima captura de pantalla.

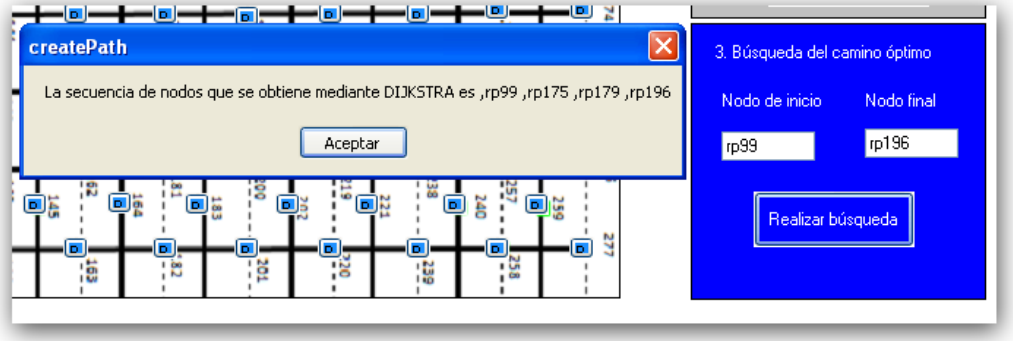

Figura 85. Solución del algoritmo Dijkstra

A continuación, se muestra el diagrama de flujo de la programación realizada para resolver la creación de un nuevo "Layout". No se entrará en detalles que correspondan a otras aplicaciones, puesto que se llevarán a cabo diagramas de flujo para cada una de ellas en sus respectivos apartados.

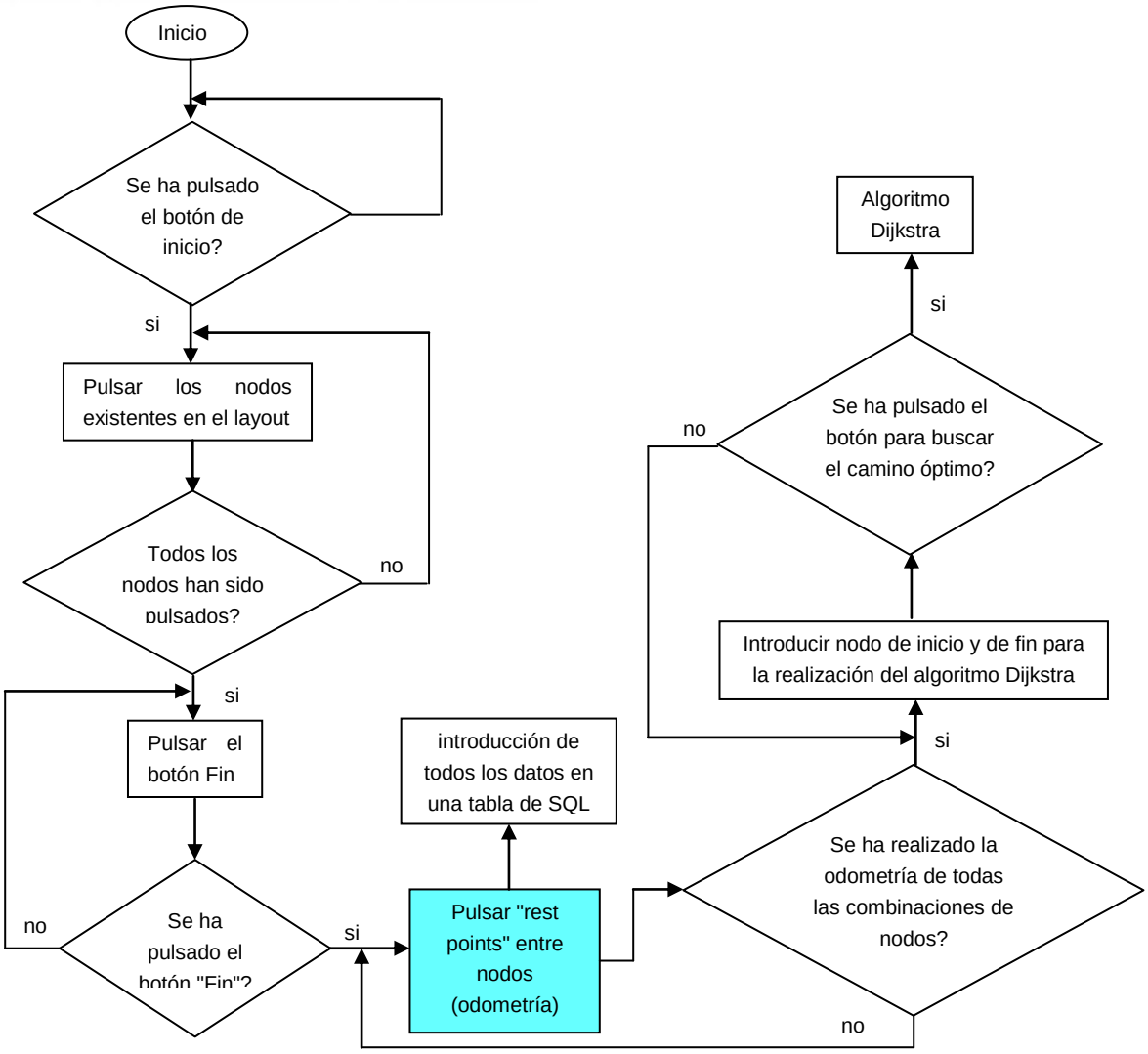

Figura 86. Diagrama de flujo de la aplicación de creación de un layout

Se puede desglosar aún más el bloque 'Pulsar "rest points" entre nodos (odometría)' pintado de color azul. Se realiza el desglose en la siguiente figura:

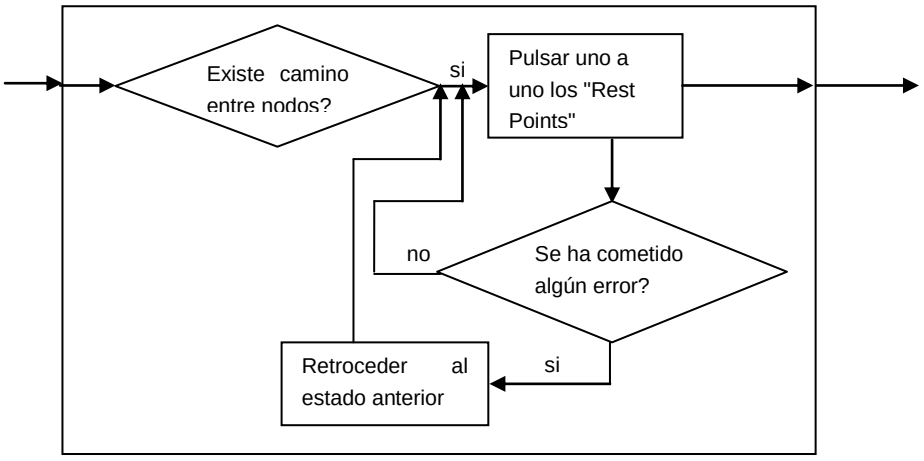

Figura 87. Bloque 'Pulsar "rest points" entre nodos (odometria)

- **Formulario para utilización del "layout" predefinido.**

Este formulario consiste en la segunda variante de la aplicación. Para acceder a dicho formulario es necesario pulsar el botón que se marca a continuación en el menú de selección de "layout".

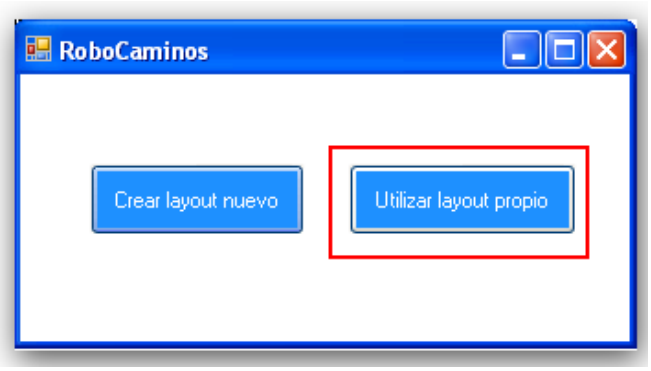

Figura 88. Selección de "layout" propio

Una vez dentro del formulario, se podrá observar la interfaz que se muestra a continuación.

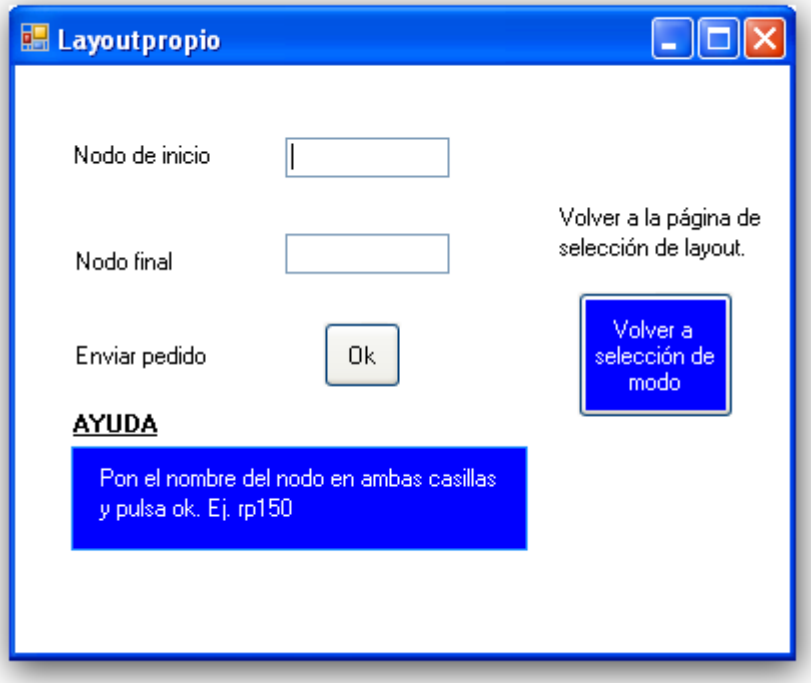

Figura 89. Formulario de utilización de "layout" propio

Como puede apreciarse, esta segunda interfaz no es tan visual como lo era la primera parte de la aplicación. Esto sucede puesto que el visualizador de dicha interfaz se puede observar en SCADA y será explicado con detalle en próximos apartados de esta misma unidad.

Existen únicamente cuatro objetos mediante los cuales el usuario es capaz de interactuar. Estos son los nodos de inicio y fin, el botón de vuelta a la página de selección de modo (realiza la misma función que en el formulario descrito anteriormente) y el botón de enviar pedido.

Además este formulario también incluye un único mensaje de ayuda para ahorrar equivocaciones al usuario. No obstante, se ha añadido en la distribución en planta que se muestra por pantalla el número de "rest point o "punto de descanso" correspondiente a los puntos críticos del "layout", como son almacenes y puntos de salida y entrada de productos. Puede observarse en profundidad este hecho en el ejemplo desarrollado en la unidad 13.

Lo que sucede al apretar el botón "Ok" será explicado en los próximos apartados de esta unidad, puesto que implica la incorporación del resto de programas utilizados en la simulación . Puede observarse en la figura 90 el diagrama de flujo que sigue la aplicación que utiliza una distribución en planta predefinida.

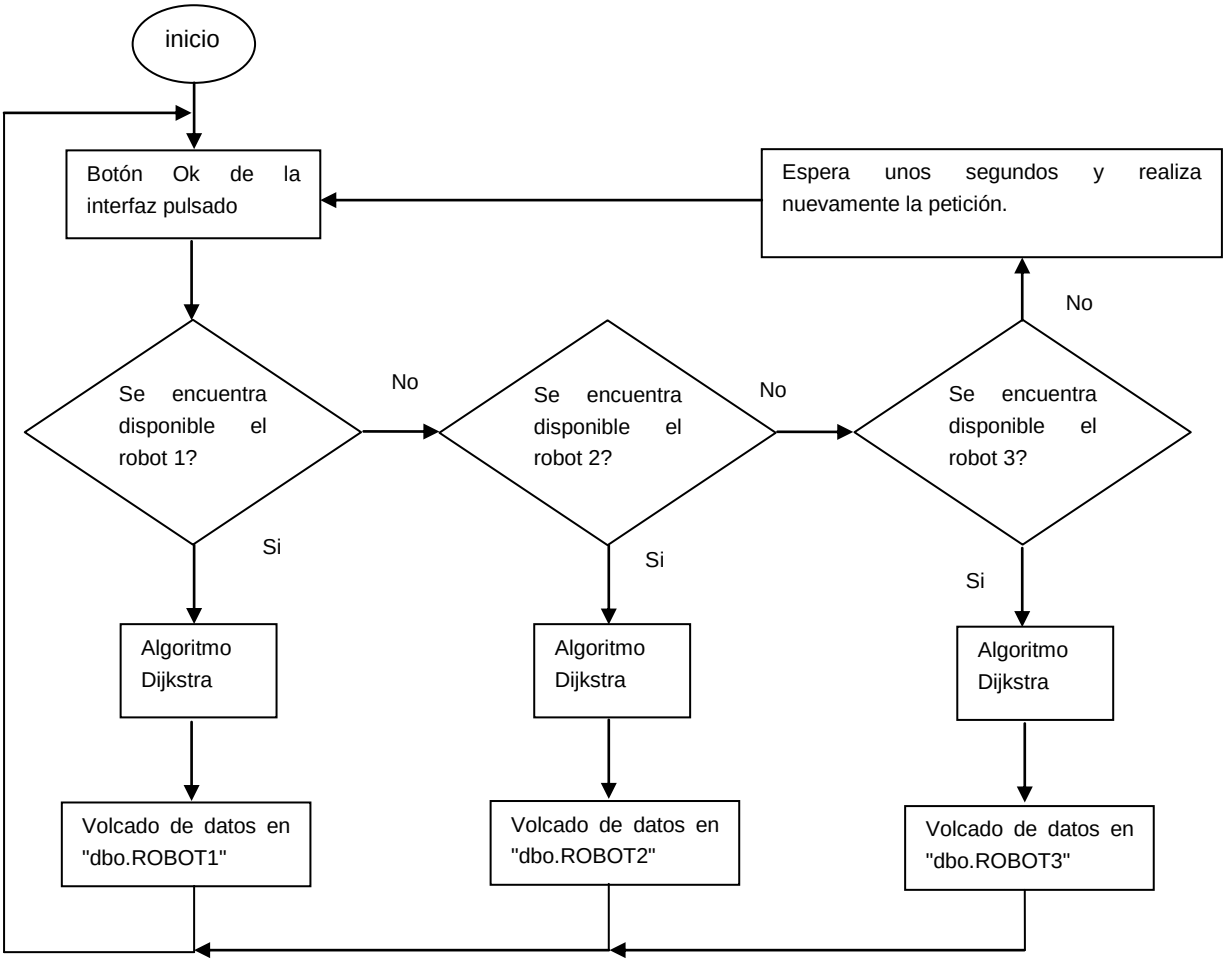

Figura 90. Diagrama de flujo de la aplicación de utilización de un layout predefinido

Tal y como puede observarse en el diagrama de flujo de la figura 90, al pulsar el botón de envío de pedido se realiza una pregunta al sistema. Primeramente se chequea si el robot 1 se encuentra disponible, es decir, sin carga. Si el robot 1 se encuentra disponible se le asigna la tarea, mientras que si se encuentra realizando un pedido entrado con anterioridad, se realiza el chequeo de disponibilidad para el segundo robot. Se plantea la misma lógica para el segundo robot, en caso de estar éste disponible se ocupará de la tarea, si se encuentra ocupado se realizará el chequeo para el tercer robot. Finalmente, si el tercer robot se encuentra disponible, será éste el que realice la búsqueda y entrega del producto. En caso de que todos los robots estén ocupados, aparecerá por pantalla el mensaje de error que se muestra en la figura 94 y que insta al usuario a volver a intentarlo pasados unos segundos.

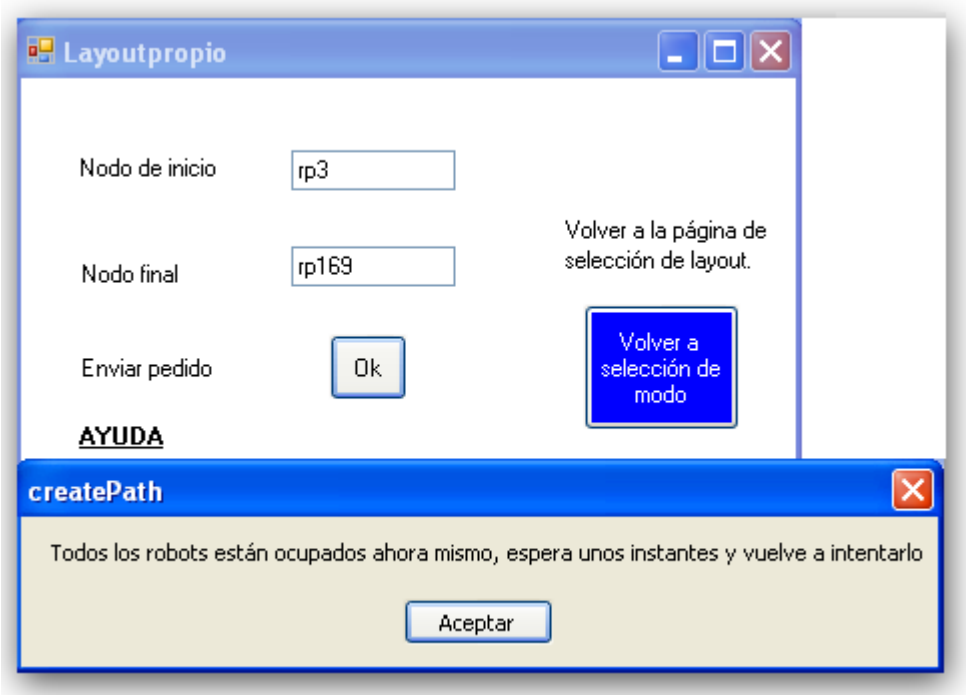

Figura 91. Mensaje de error, robots ocupados

11.1.3. Restricciones aplicadas al sistema

Este apartado hace referencia a las tareas que el sistema no permite realizar, programablemente hablando. El formulario de la aplicación que contiene dichas restricciones para el usuario es la primera expuesta anteriormente, la interfaz para la creación de un nuevo "layout".

Empezando des del inicio de la simulación, el usuario ya encuentra diversas restricciones. La aplicación se encuentra en la situación correspondiente a la figura 69, ahora el usuario debe pulsar inicio y empezar con la selección de nodos. Si el usuario no pulsa inicio y trata, seguidamente, de pulsar fin, recibirá el mensaje de error que se muestra a continuación.

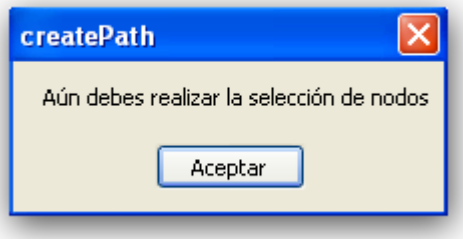

Figura 92. Falta realizar la selección de nodos

Si en vez de eso el usuario trata de pulsar cualquier otro botón de otros bloques, que ahora se encuentran inactivos, recibirá la siguiente advertencia.

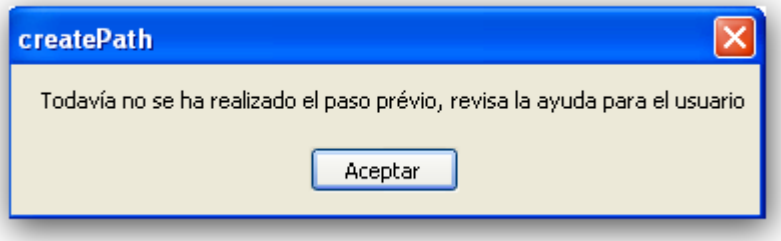

Figura 93. Falta realizar el paso previo

Por otro lado, pero siguiendo en el punto en que el usuario todavía no ha pulsado inicio, si se pulsan "rp" estos se marcaran de verde pero al pulsar el botón de inicio las tablas se vaciaran automáticamente.

Se recibirán los el mismo mensaje de error que el que se muestra en la figura 88, en caso de pulsar inicio y sin haber pulsado fin se trata de realizar los siguientes pasos.

El próximo paso es comprobar que sucede cuando la interfaz del programa ha llegado al segundo bloque, si estando en la reproducción de la odometría (correspondiente a las figuras 74 y 75) se pulsa alguno de los botones del primer bloque se recibirá el siguiente mensaje de error.

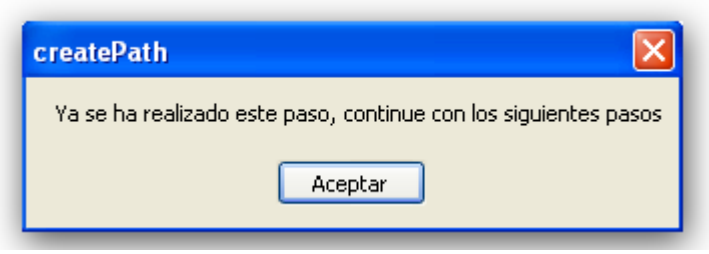

Figura 94. Error de paso realizado

En caso de pulsar el botón que realiza la búsqueda del algoritmo Dijkstra, se recibirá el mismo mensaje de error que el mostrado en la figura 88.

Finalmente, una vez se ha llegado al tercer bloque donde únicamente queda por realizar el Dijkstra, si el usuario pulsa cualquier botón del segundo bloque, recibirá el siguiente mensaje de error.

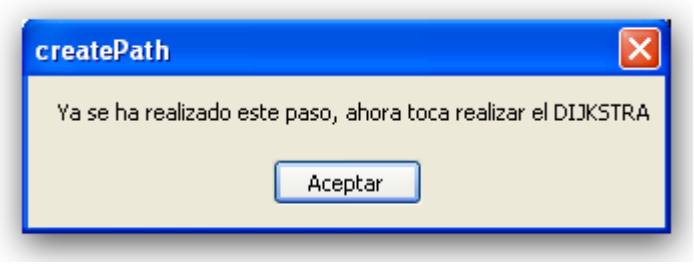

Figura 95. Error para realizar Dijkstra

Si por lo contrario, en ese momento, se pulsa cualquiera de los botones que se encuentran en el primer bloque, se continuará recibiendo el mismo mensaje de error que el mostrado en la figura 88.

Para finalizar, la última restricción que se recibirá se puede reproducir mientras se está realizando el último paso del programa (realización de la búsqueda del comino óptimo). La restricción se encuentra al pulsar cualquiera de los 276 "rest points" que se muestran en el "layout", y el mensaje de error que se recibirá es el que se puede observar a continuación.

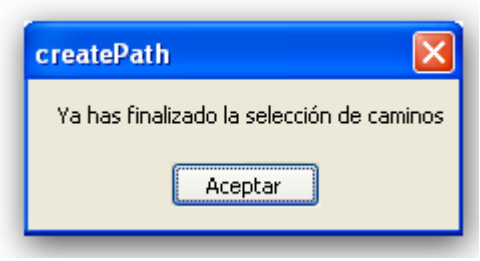

Figura 96. Error al haber finalizado la selección de caminos

11.2. Base de datos SQL Server

Se organizará este apartado en tres sub-apartados que cubrirán la estructura interna de tablas que utiliza el programa, pasos para la creación de una base de datos y sus tablas y finalmente se realizará una explicación de que son y cómo se han utilizado en este proyecto los procesos almacenados.

11.2.1. Estructura de tablas

Primeramente, se realiza a continuación una muestra por encima de la distribución de tablas dentro de SQL, puede observarse dicha distribución en la ilustración 92.

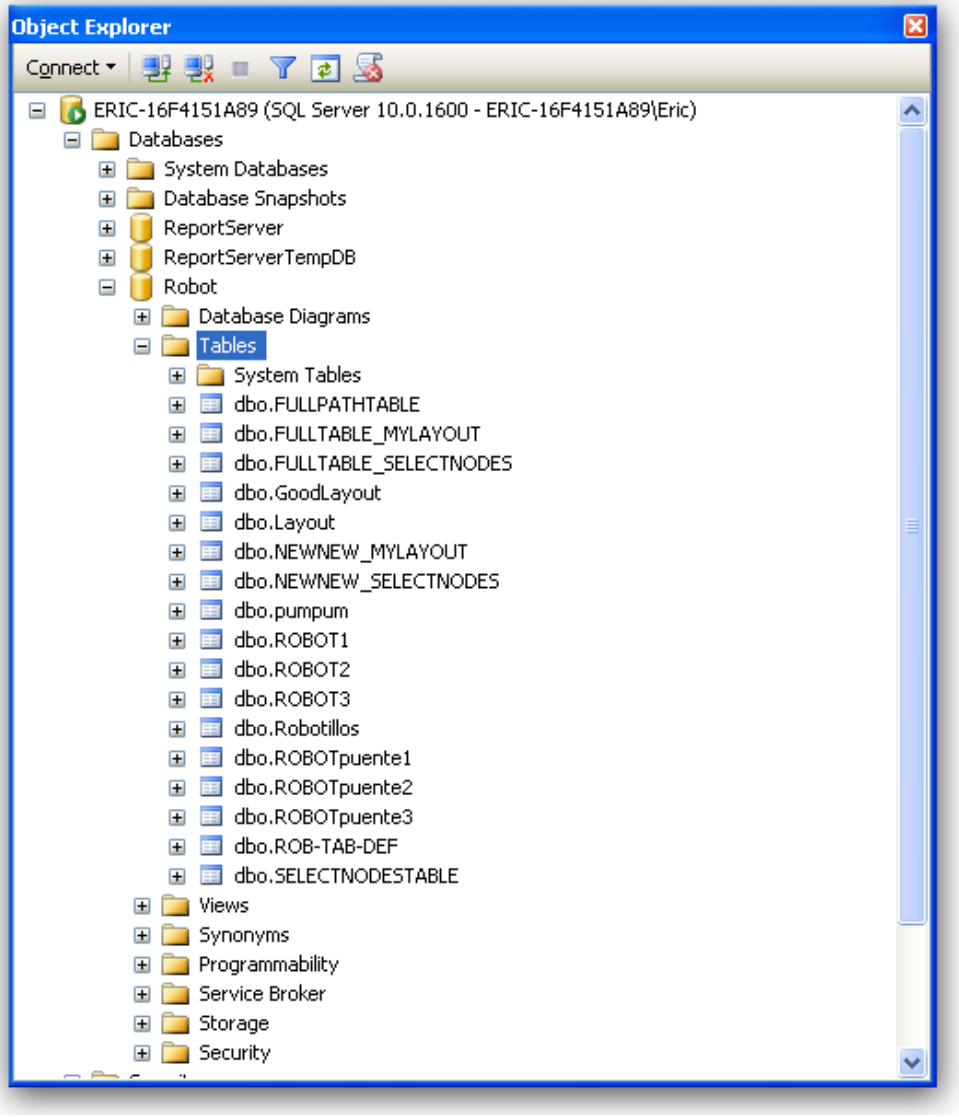

Figura 97. Estructura de tablas dentro de la base de datos SQL

Para realizar una explicación detallada de cómo está estructurado el sistema en lo relativo a la base de datos SQL que se está utilizando, se debe empezar

enlazando debidamente la explicación con el apartado anterior. Es decir, hace falta constatar en qué punto empiezan a trabajar las tablas SQL y qué función tienen en el sistema.

Utilizando la interfaz de creación de layout expuesta con anterioridad se reproduce fielmente el layout que se puede observar en la figura 60. A medida que se siguen los pasos expuestos previamente para la creación de una nueva distribución en planta, se rellenan dos tablas distintas. La primera guarda todas y cada una de las posiciones que el robot tendrá que tomar, mientras que la segunda realiza la correspondencia entre cuatro valores, los nodos de inicio y fin y las filas de inicio y fin que comprende en la anterior tabla el tramo que se está recorriendo. De esta forma se rellenaron las tablas "dbo.NEWNEW_MYLAYOUT", que corresponde a la tala que almacena la odometría que seguirán los robots, y la tabla "dbo.NEWNEW_SELECTNODES", que guarda la correspondencia entre la tabla anterior y los nodos del grafo.

A partir de este punto, entra en acción la lógica que está detrás del diagrama de flujo que se muestra en la figura 90, esta sigue el siguiente flujo (donde A = dbo.NEWNEW_MYLAYOUT y B = dbo.NEWNEW_SELECTNODES) :

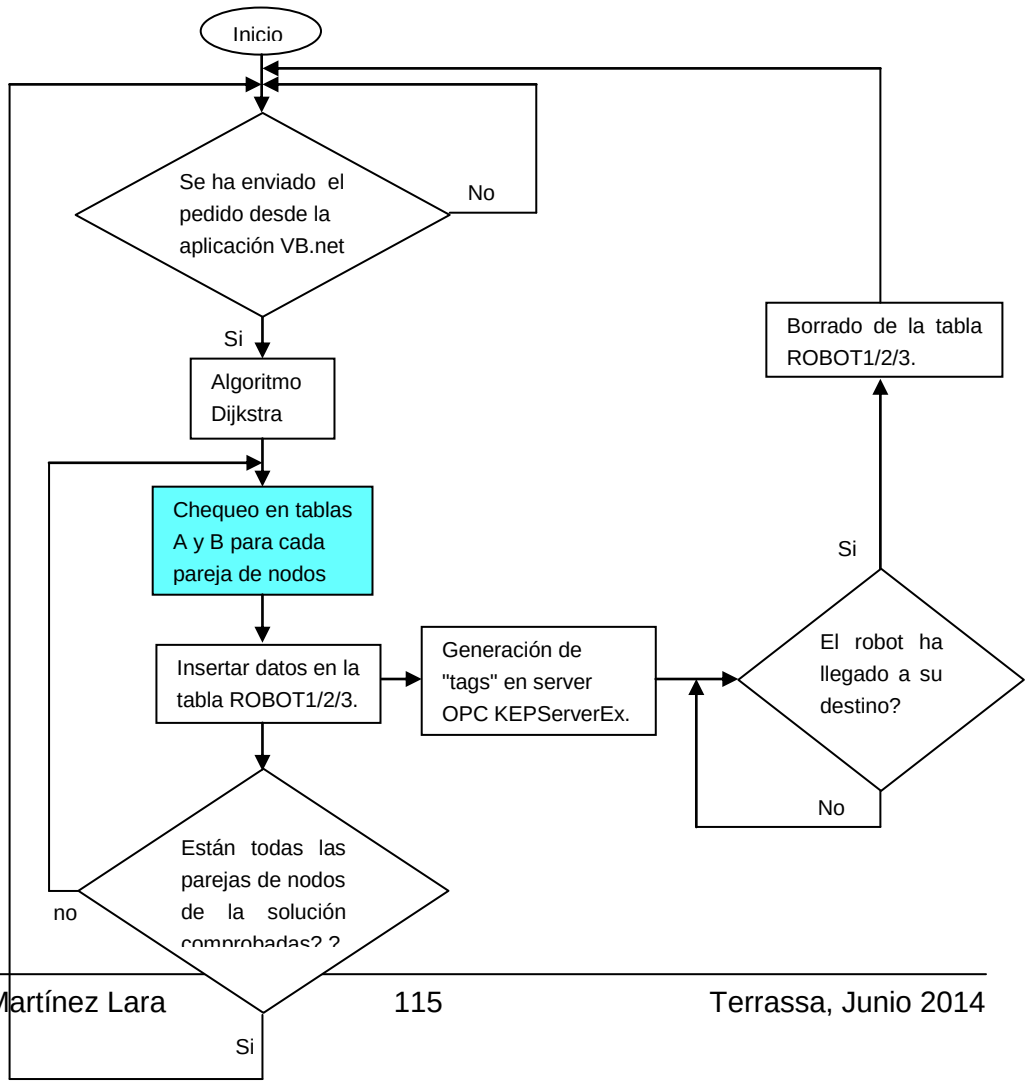

Figura 98. Diagrama de flujo de la lógica de la programación SQL

Mirando la figura 97, se ha marcado un bloque de color azul claro, puesto que dicho bloque, puede desglosarse en otros más específicos, tal y como se muestra en la imagen que hay a continuación. (donde A = dbo.NEWNEW_MYLAYOUT y B = dbo.NEWNEW_SELECTNODES)

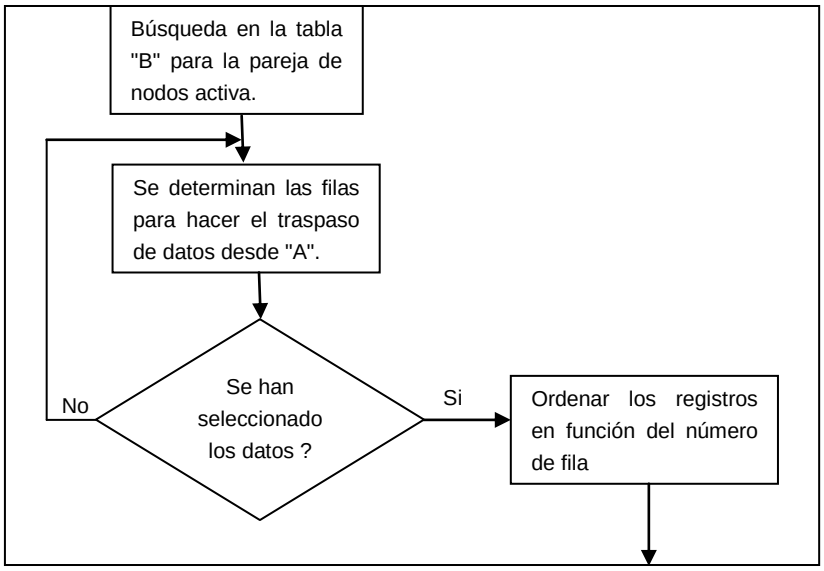

Figura 99. Bloque 'Chequeo en tablas A y B para cada pareja de nodos' Ampliado

Estas tablas se refrescan una vez los robots han llegado a la posición final. Este refresco de las tablas, se hace mediante el enlace de un cliente OPC llamado ClientACE y el programa Visual Basic que será comentado en próximos apartados de este capítulo.

El resto de tablas se han construido para dar soporte a la infraestructura interna de tablas, se utilizan como puente de conexión entre otras tablas o para guardar información instantánea que al ser transportada a otra tabla posteriormente será eliminada.

11.2.2. Creación de tablas dentro de una base de datos SQL

El primer paso a realizar para la creación de una serie de tablas en el entorno de SQL Server es la creación de una base de datos.

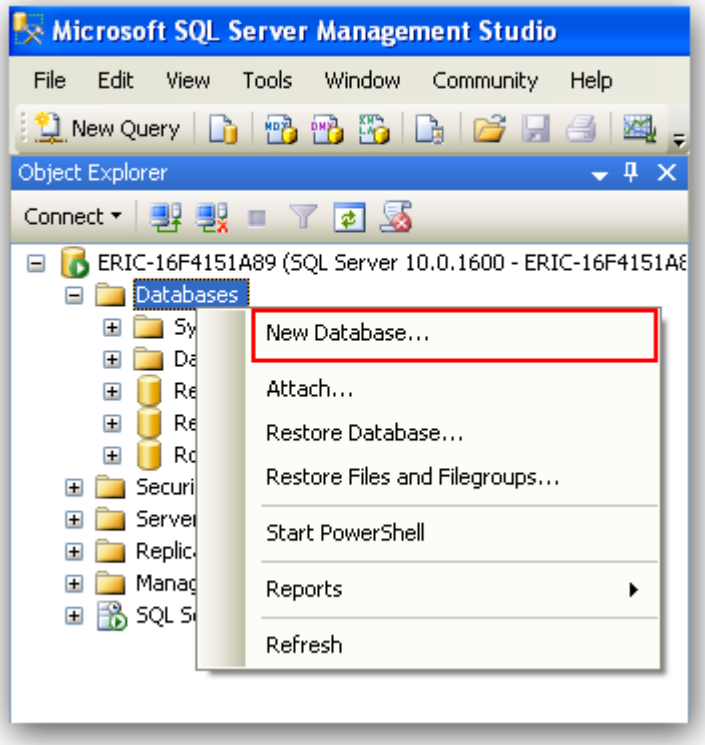

Figura 100. Creación de una base de datos SQL Server

Al pulsar "New Database" o Nueva Base de datos aparecerá por pantalla el formulario en el cual únicamente se debe rellenar el nombre y posteriormente se debe pulsar el botón Ok. Al realizar esto, la nueva base de datos aparecerá en la lista desplegable.

Una vez la base de datos ha sido generada, existen dos posibilidades de generar tablas en SQL Server. La primera de ellas consiste en un método más intuitivo para el usuario, está pensado para personas con un dominio limitado de programación SQL. Por otro lado, el segundo método consiste en la creación de consultas, que no dejan de ser pequeños programas que generan la tabla a gusto del consumidor de forma más precisa. Más adelante se pueden observar ejemplos para ambos métodos de creación de tablas.

Empezando primeramente con el método sencillo de creación de tablas, al pulsar con el botón derecho sobre las tablas de la base de datos creada, aparecerá un desplegable que nos ofrece la opción de generar una tabla.

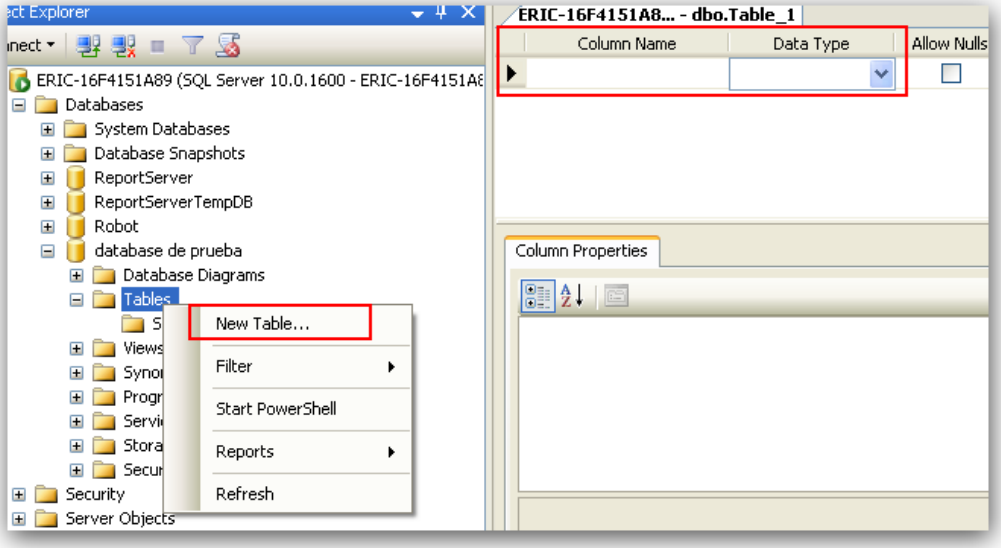

Figura 101. Método sencillo para creación de tablas SQL

Tal y como puede observarse en la captura anterior, al añadir una nueva tabla te permite tres opciones distintas para dar formato y personalización a la tabla. El primer campo sirve para indicar el nombre de la columna que deseas añadir, el segundo campo se utiliza para describir le tipo de dato que se va a insertar en esa columna, tal y como puede observarse en la anterior imagen, el programa ofrece un desplegable para que se pueda seleccionar el tipo de dato deseado. Por último debe indicarse en la última casilla si se permiten valores nulos en cualquiera de los registros para esa columna. Por ejemplo, si se desease insertar en una tabla una columna para guardar los días de la semana, y otra columna para guardar el dinero gastado en cada uno de esos días, se ejecutaría la tabla tal y como se muestra en la figura 102 y posteriormente se grabaría.

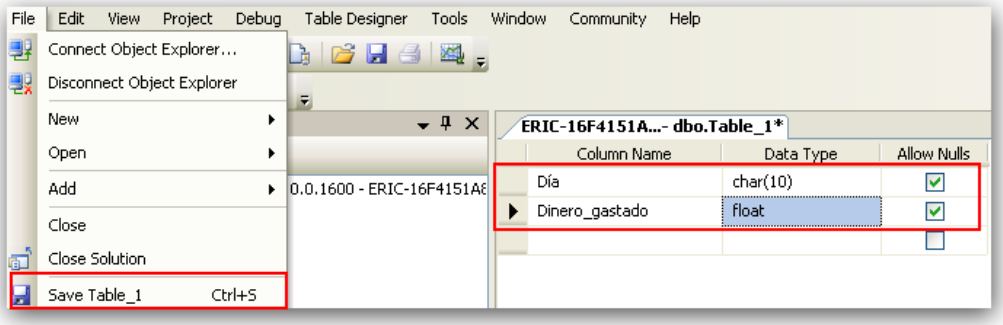

Figura 102. Ejemplo método sencillo para creación de tablas SQL

Por otro lado, pasando al método complejo, obviando que se ha de poseer unas mínimas nociones básicas de programación SQL, en la figura que se muestra a continuación se muestran como generar una tabla utilizando dicho método.

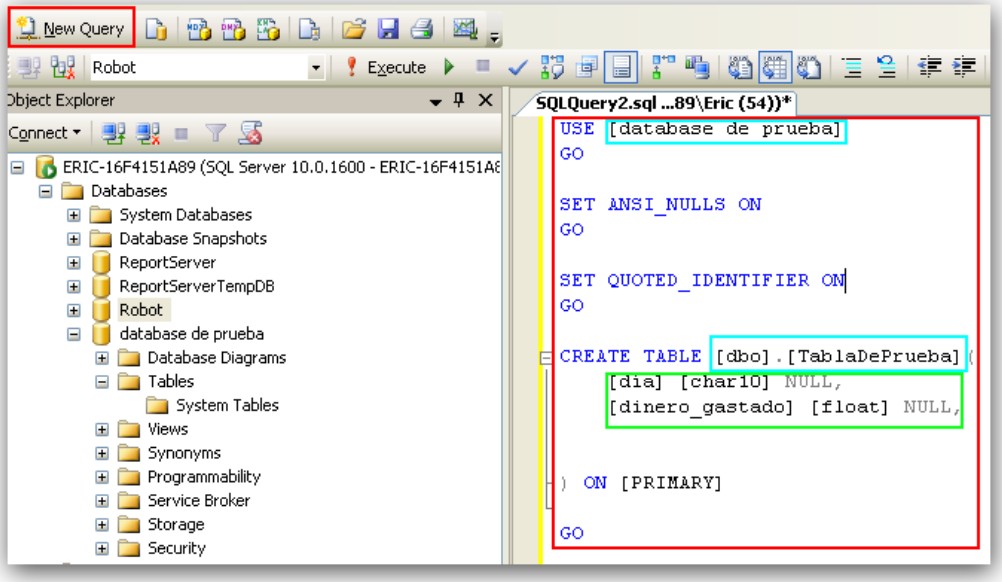

Figura 103. Método complejo de generación de tablas en SQL Server

Primeramente se debe generar una nueva consulta o "New Query", aparecerá una nueva pestaña en blanco en la que escribir el código. Las partes más importantes a tener en cuenta las partes del código que han sido marcadas en la figura anterior. Los recuadros en azul son tanto el nombre de la base de datos en el que se introducirá la tabla, cómo el nombre que deseas que la tabla posea al generarla, en este caso será "TablaDePrueba". Posteriormente, con el mismo formato que se utilizaba en el método sencillo de creación de tablas, se añaden las columnas que se desea tener en dicha tabla. En este caso se han utilizado las mismas columnas que se presentaban en el ejemplo sencillo expuesto anteriormente.

La condición para que dicha tabla se genere es pulsar el botón de la interfaz que se muestra marcado en la captura de pantalla que hay a continuación.

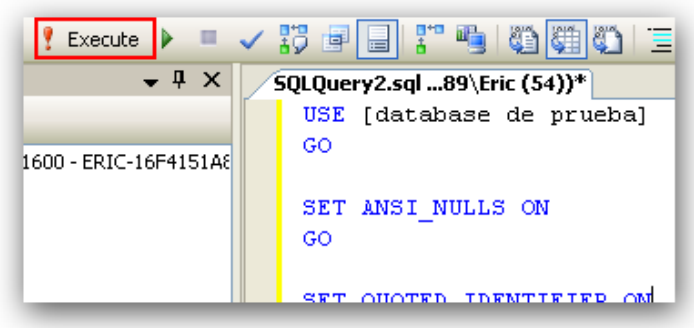

Figura 104. Ejecutar para que la tabla se genere en el sistema

11.2.3. Procesos almacenados

Esta sección ilustra que son y cómo se trabaja con los procesos almacenados de SQL dentro de este proyecto.

Un proceso almacenado no es más que un código de SQL preparado que se guarda para poder reutilizar tantas veces como se requiera. De esta forma, si se piensa en una consulta que se escribe una y otra vez, como por ejemplo el volcado de datos desde la tabla maestra a cada uno de los robots. Así pues, en vez de tener que escribir una consulta por cada vez que se necesite realizar dicho volcado de información, se guardaría este código en un proceso almacenado de manera que únicamente habría que llamarlo desde el programa padre para que se ejecute. De esta forma, no solamente se mejora el rendimiento del sistema, sino que además es más sencillo realizar actualizaciones de estos procesos que de un código mucho más extenso. Por otro lado, estos procesos almacenados ofrecen la posibilidad de pasar parámetros a dichos procesos, así que dependiendo en cual sea la necesidad del programa en cada instante, el proceso almacenado puede actuar en base a los valores del parámetro que le fue introducido.

En este proyecto, se han utilizado tres procesos almacenados que son los encargados de realizar, tal y como se ha comentado antes, el volcado de la tabla de datos maestra hacia cada una de las tablas de los robots. En función del robot que vaya a realizar la tarea, se hará la llamada a un proceso almacenado o a otro. En la próxima captura de pantalla se muestran los procesos almacenados que han sido generados para la realización de este proyecto.

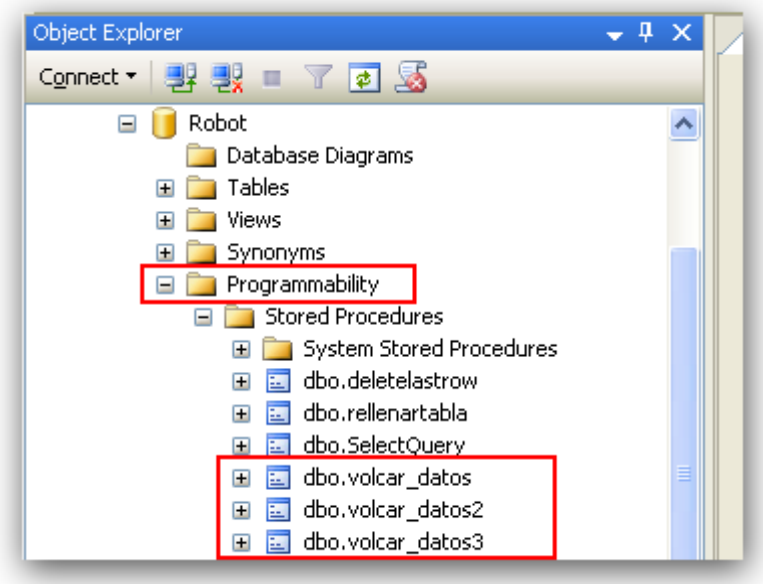

Figura 105. Estructura de procesos almacenados

Los procesos que han sido seleccionados en color rojo son los que están siendo utilizados actualmente en el sistema.

Por otro lado el sistema ofrece una amplia ayuda al usuario a la hora de crear estos procesos almacenados. Para crear dichos procesos, primeramente hay que clicar con el botón derecho sobre procesos almacenados, que se encuentra dentro del apartado "Programabilidad".

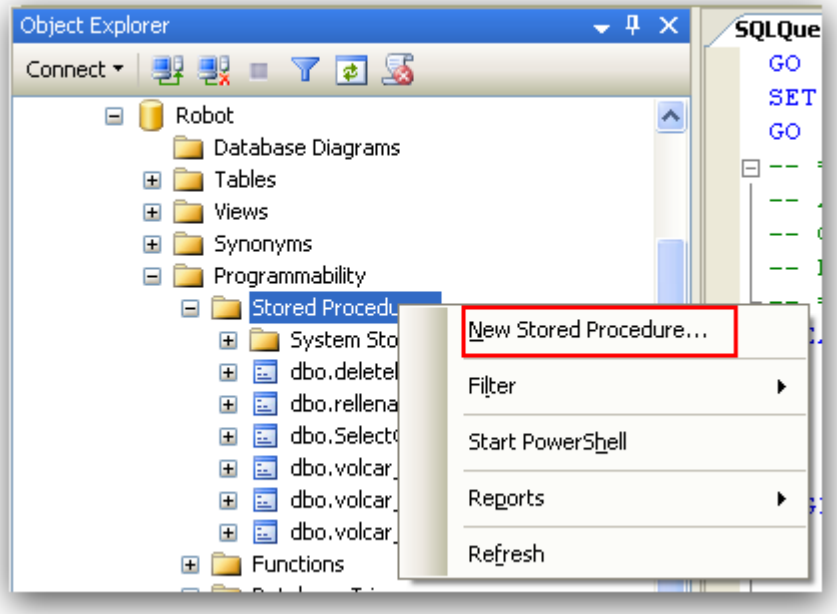

Figura 106. Creación de un proceso almacenado

Una vez generado el nuevo proceso almacenado, aparece una nueva pestaña o consulta con una plantilla predeterminada por el programa para facilitar la creación del nuevo proceso. Se puede observar dicha consulta en la figura 107.

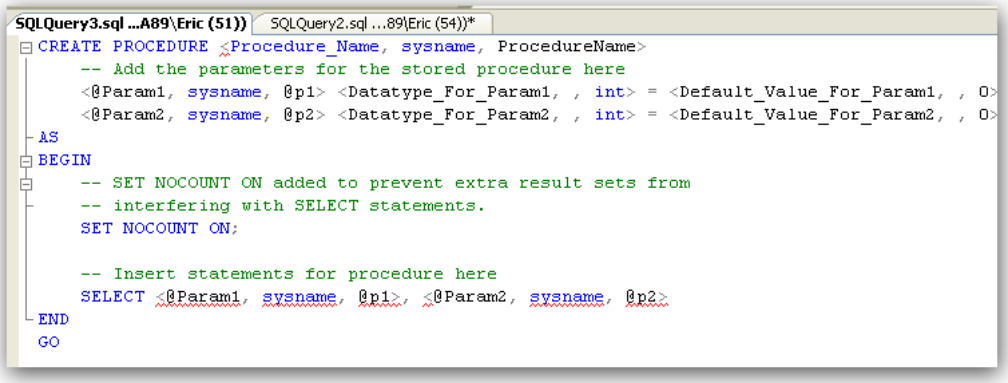

Figura 107. Consulta de creación de un proceso almacenado

Una vez finalizado el código, nuevamente solamente hace falta presionar el botón de ejecutar para que el sistema lo guarde como un nuevo proceso almacenado.

Posteriormente, solo queda aprender el método para llamar estos procesos almacenados desde la lógica de la aplicación base. En el caso de este proyecto, desde Visual Basic, se mostrará cómo se desarrolla esta llamada desde en el próximo capítulo.

Los procesos almacenados en este proyecto tienen el aspecto que se observa en la figura 108, la única diferencia es el nombre, y la tabla en la que se insertan los valores.

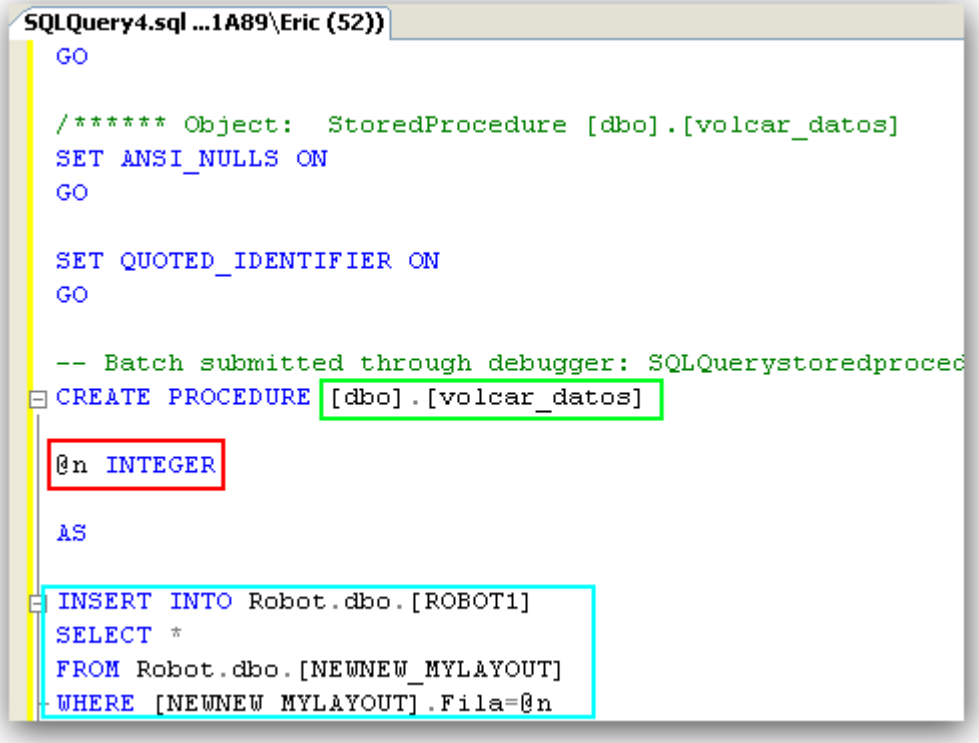

Figura 108. Proceso almacenado volcar_datos

Enmarcado en color verde se puede observar el nombre que tendrá el proceso almacenado, en color rojo se distingue el parámetro en base al cual realizará el volcado el proceso y finalmente en azul está la lógica que ejecuta el proceso cuando este es llamado desde el programa. Básicamente, inserta un registro en la tabla ROBOT1 desde la tabla maestra NEWNEW_MYLAYOUT donde la columna fila de dicha tabla tenga el valor del parámetro que se le está introduciendo al proceso des del programa.

11.3. Archivo de servicio OPC

En este apartado se abordará el funcionamiento del servidor OPC KEPserverEX 5.0, a la vez que se hará un estudio de algunos de los drivers que este servidor ofrece. Mientras que el cliente OPC ClientACE y será tratado en otros capítulos.

11.3.1. Funcionamiento de KEPserverEX 5.0

Tal y como se ha comentado con anterioridad, el servidor OPC KEPserverEX tiene la función de emular el movimiento que tendrán los tres robots. Se puede observar la estructura de dispositivos que se ha utilizado para realizar este proyecto en la figura que se muestra a continuación.

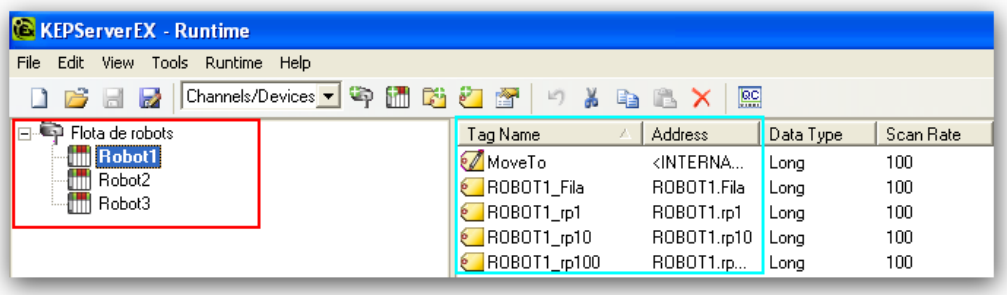

Figura 109. Estructura de dispositivos en KEPServerEx

Tal y como se puede comprobar en la imagen anterior, se ha generado un canal de dispositivos al cual se ha nombrado flota de robots, y dentro de este canal se han dado de alta tres dispositivos que corresponden a los tres robots que irán realizando la entrega de pedidos. Cada uno de los robots, tal y como puede observarse en la zona encuadrada en color azul, tendrá tantos "tags" o "etiquetas" como posiciones pueda llegar a tomar (o en otras palabras, tantos "tags" como columnas tenga la tabla que enlacemos). En este caso serán 276 "tags" puesto que las tablas enlazadas con cada uno de los dispositivos tienen 276 campos correspondientes a los 276 "rest points" del "layout".

A continuación, pese a ser un proceso relativamente sencillo, se muestra cómo generar canales y dispositivos en KEPServerEx. En primer lugar es necesario pulsar el botón marcado en rojo en la figura 110 para dar de alta un nuevo canal.

Figura 110. Botón para la creación de nuevos canales

Una vez pulsado dicho botón aparecerá por pantalla el siguiente formulario:

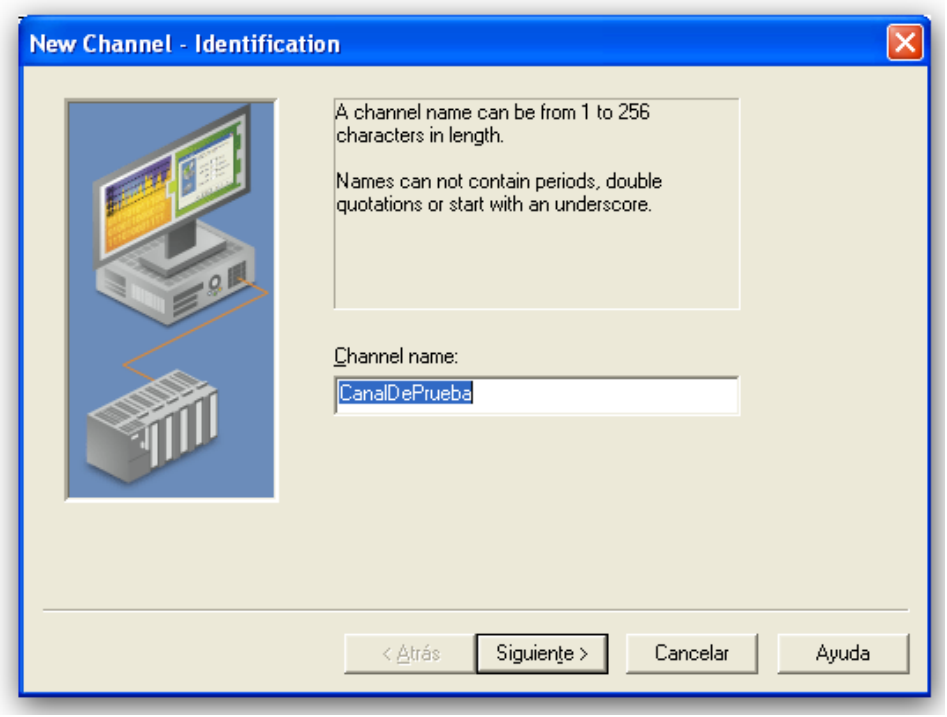

Figura 111. Nombre del nuevo canal

Un vez introducido el nombre del canal, la aplicación requiere seleccionar un driver de toda una lista de drivers que esta proporciona. Tal y como se muestra en la figura siguiente.

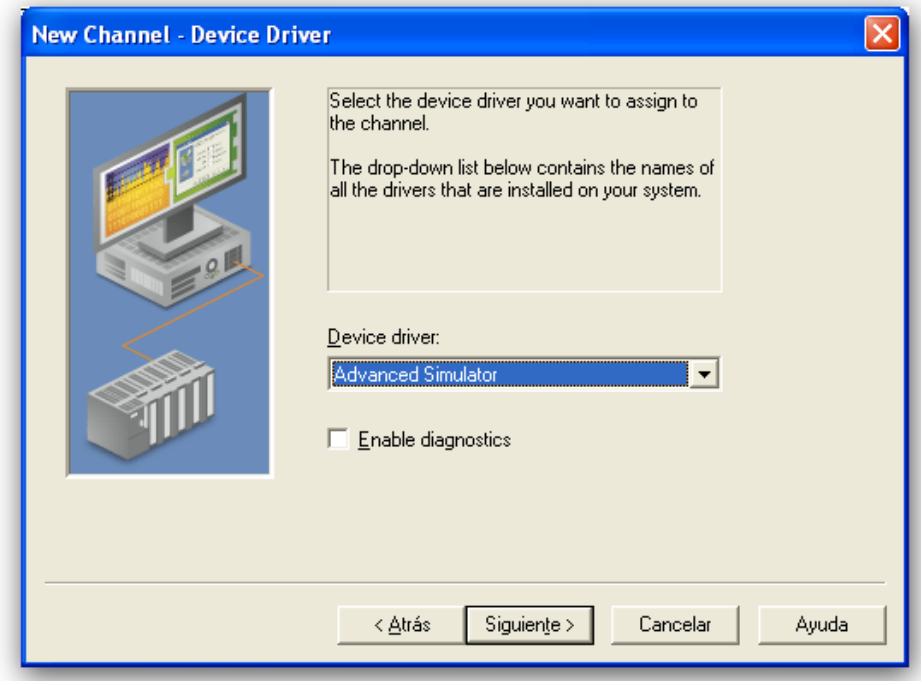

Figura 112. KEPServerEx selección de driver

Se realizará un estudio de los principales drivers disponibles y del driver finalmente seleccionado en el próximo apartado. Por ahora, únicamente añadir que se ha escogido el driver Advanced Simulator por su afinidad con bases de datos SQL, ya que esto proporcionaba al sistema una estructura funcional más sencilla.

De los siguientes pasos después de la selección del driver no hace falta mencionar prácticamente nada más, debido a que no debe modificarse la opción que se da por defecto. Por otro lado, en el proceso de creación del canal, después de seleccionar el driver Advanced Simulator, el sistema pide añadir una fuente de datos, pero esto será tratado en la próxima unidad. Una vez rellenados todos los datos necesarios, se añadirá un nuevo canal al servidor OPC tal y como se muestra en la siguiente figura.

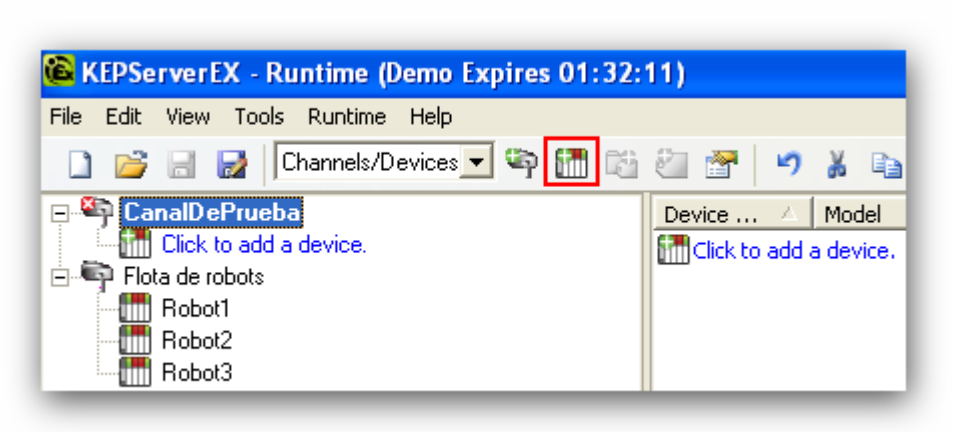

Figura 113. Creación de canal finalizada

El siguiente paso es añadir los dispositivos, en el caso de este proyecto, los dispositivos son los emuladores de los robots. Con el fin de añadir un nuevo dispositivo a el canal que se ha creado se debe pulsar el botón que se ha marcado en rojo.

Una vez se pulsa dicho botón aparecerá en pantalla un nuevo formulario, dicho formulario pedirá al usuario un nombre para el nuevo dispositivo que se dispone a crear, se muestra un ejemplo en la figura 114.

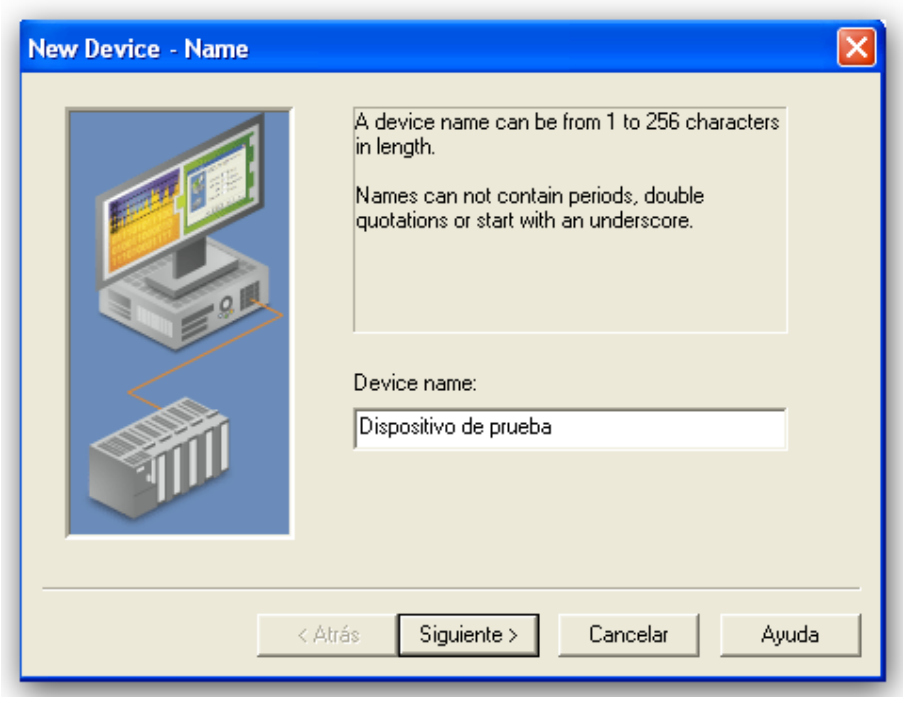

Figura 114. Generación de un dispositivo

De entre todos los pasos que el programa pide seguir a continuación, es importante hacer mención al tiempo de refresco de los datos de la taba. En un momento dado, se requiere introducir cada cuanto tiempo se realizará el cambio de registro dentro de la tabla pasando al inmediatamente siguiente.

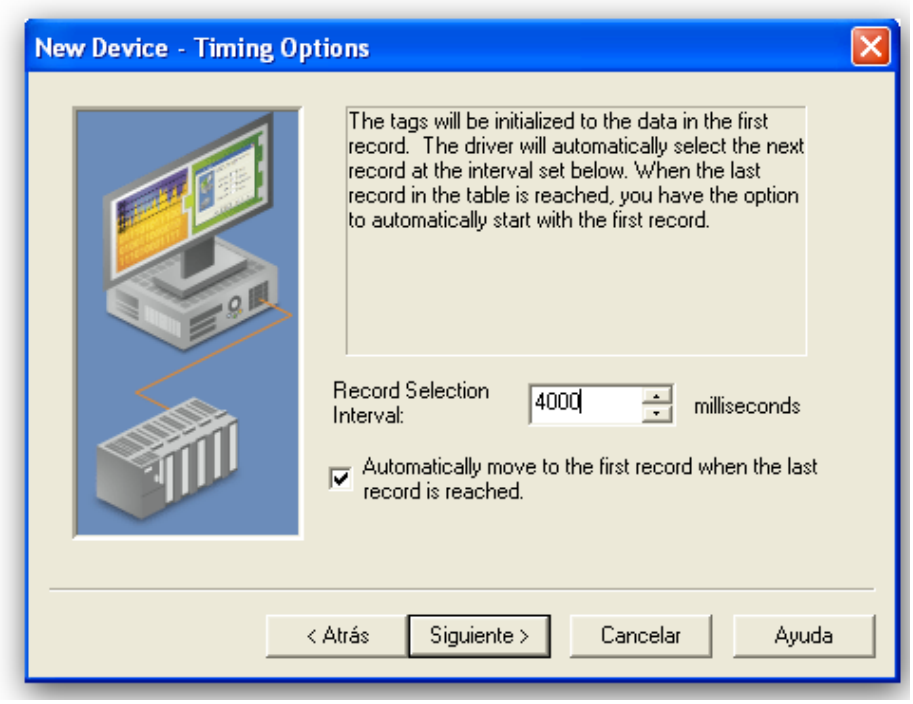

Figura 115. Opción de tiempo de refresco de la tabla

Se ha seleccionado un tiempo de 4000 milisegundos, lo que equivale a 4 segundos, debido a que la distancia entre cada "rest point" es de 4 metros y el robot viaja a una velocidad de 4 m/s. Así pues, el tiempo que tardará el robot en modificar su posición de un "rest point" a otro, será de 4 segundos.

Finalmente se ofrece la opción de volver al primer registro de la tabla una vez se ha llegado al último, en el caso de este proyecto, era relevante volver a reiniciar el movimiento del robot en cuanto la tabla volviese a rellenarse, por ese motivo se dejó marcada esa opción.

Una vez se ha generado el dispositivo, este aparecerá dentro del canal y se generarán los "tags" en función de la tabla escogida, puede observarse un ejemplo en la figura 116. Nuevamente, es necesario recordar que el proceso de enlace entre SQL y KEPServerEx se desarrollará detalladamente en la próxima unidad.

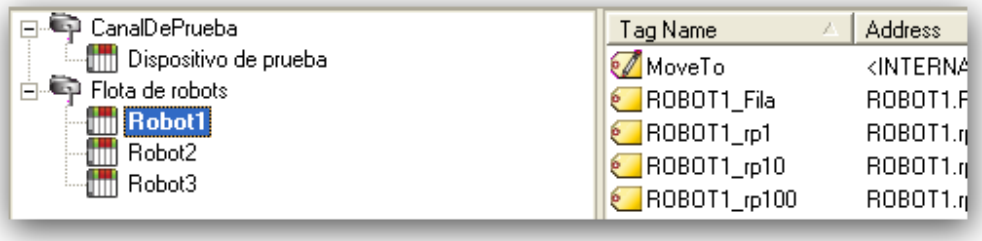

Figura 116. Creación de un dispositivo dentro del canal

11.3.2. Drivers estudiados

En este apartado se realizará un estudio de los drivers estudiados que mejor encajaban con el proyecto en cuestión, finalizando con el driver finalmente seleccionado. La instalación y el uso del driver seleccionado se tratará en el próximo capítulo, aquí únicamente se mostrará un visión general del funcionamiento de dicho driver.

El primer requisito para la preselección de drivers ha sido que sean drivers de libre acceso y utilización, es decir, que no sean drivers de propietario. Por lo tanto drivers de marcas como OMRON, Toshiba, Siemens, etc. han sido descartados.

La estructura de presentación de dichos drivers, será distribuida en dos apartados. Primeramente se realizará una descripción del driver y en segundo lugar se tratará el proceso de configuración del mismo.

Driver DDE Client

- **Descripción**

El DDE Client Driver provee una forma sencilla y fiable para conectar clientes DDE con aplicaciones OPC Cliente, incluyendo HMI, Scada, Historian, MES, ERP y muchas más aplicaciones.

Un servidor OPC puede proporcionar un único punto para administrar múltiples servidores DDE mediante la velocidad y potencia OPC.

Una de las características de los drivers DDE Client es que son capaces de comunicarse con cualquier server que soporte el estándar de formato de datos CF TEXT DDE . Acepta clientes remotos y comunicaciones con este driver vía OPC o NetDDE. Sin embargo, la versión actual de este driver no permite acceder a datos que estén localizados en servidores DDE trabajando en máquinas remotas.

- **Configuración**

El primer paso de configuración de este tipo de servidores es seleccionar el mismo a la hora de crear un nuevo canal, tal y como se muestra en la siguiente figura.

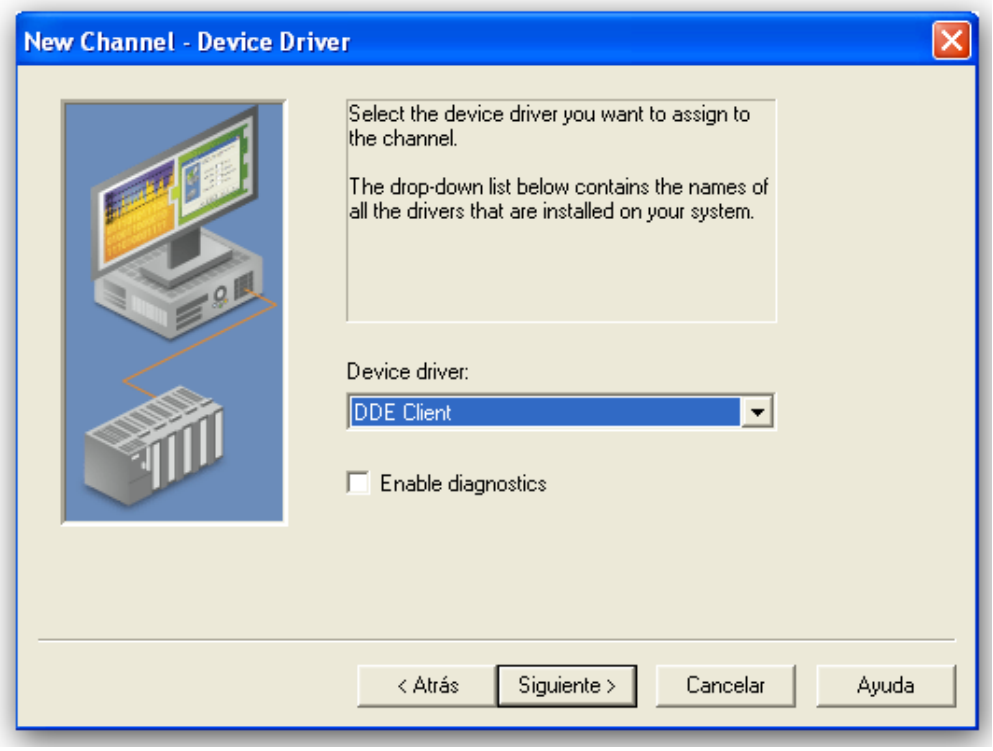

Figura 117. Selección del driver DDE Client

Para el caso de un DDE Client Driver, el dispositivo en el proyecto de OPC servidor no tiene porque corresponder necesariamente a un dispositivo físico en una red. De hecho, el dispositivo es un programa DDE cliente que puede recolectar datos que provengan de distintos servidores DDE. Además, aunque haya configurado un solo dispositivo dentro del canal, éste puede ser utilizado para comunicarse con cualquier numero de dispositivos físicos a través de aplicaciones de servidores DDE.

Un aspecto importante a tener en cuenta es que un proyecto de servidor OPC, puede tener un único canal poseedor de un driver DDE Client, y ese canal puede tener un único dispositivo (por ejemplo un programa DDE Client). Por otra banda, un proyecto de servidor OPC puede tener canales adicionales que utilicen diferentes drivers.

Aunque los parámetros de comunicación de un servidor DDE son configurados durante el último paso de la creación del dispositivo, estos pueden ser cambiados en cualquier momento. Para llevar a cabo esta acción, hay que entrar en las propiedades del dispositivo clicando con el botón derecho del ratón en el dispositivo cuyo driver es DDE Client y entonces seleccionar "Properties" (tal y como puede observarse en la siguiente figura).

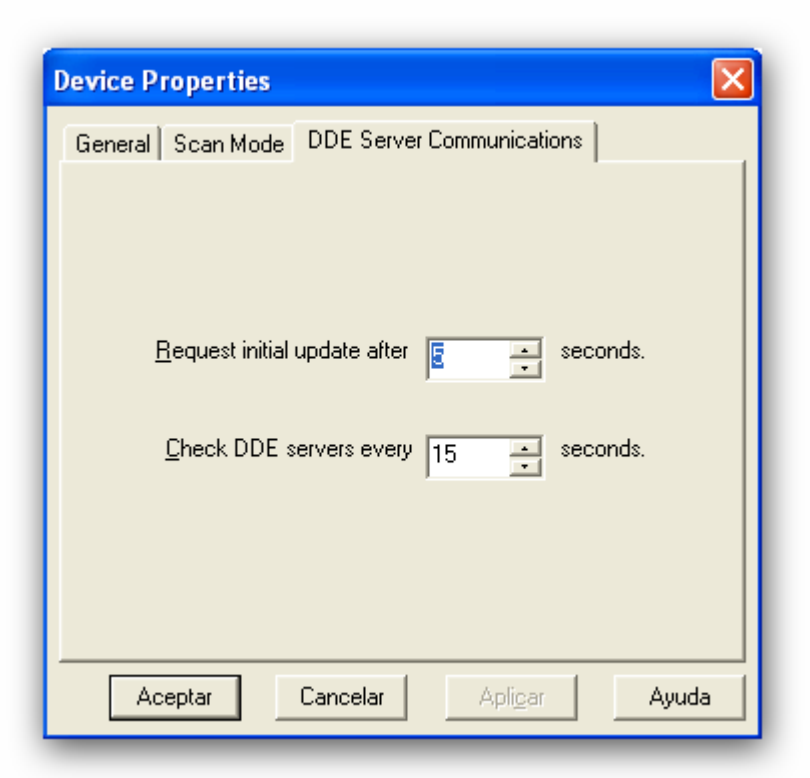

Figura 118. Propiedades de comunicación de un dispositivo DDE Client

Muchos de los servidores DDE enviarán automáticamente a sus clientes un valor actual de los objetos que incorporen tan pronto como se les haga una conexión. Para aquellos a los que no se les haga una conexión, este driver debe pedir las actualizaciones iniciales explícitamente.

El primer valor que se muestra en la figura 118 ("*Request initial update after <number> seconds*" o pide la actualización inicial despues de <n> sgundos), puede ser utilizado para controlar la cantidad de tiempo que el driver debería esperar para la actualización del vínculo inicial, antes de asegurar una demanda explicita para ello. Para este tipo de servidores, se suele sugerir un valor bastante amplio de segundos para garantizar a los enlaces suficiente tiempo para ser creados antes de que el driver empiece a pedir emisiones para actualizaciones iniciales. Para los servidores que envíen automáticamente actualizaciones iniciales cuando los vínculos son creados, cualquier valor funciona correctamente.

Este driver utiliza vínculos activos, donde un servidor DDE envía datos al driver solo cuando el valor del dato ha sido modificado. Esto provee una significante ventaja en cuanto al rendimiento sobre otras respuestas cíclicas alternativas, ya que cualquier transacción DDE consume relativamente bastante tiempo. Sin embargo, esta eficiencia tiene un precio, el driver puede no recibir noticias cuando el servidor DDE haya dejado de estar disponible (por el motivo que sea). Por esta razón, se acostumbra a utilizar el segundo valor que se muestra en la figura 118 ("*Check DDE Servers every <number> seconds*" o Comprueba los servidores DDE cada <n> segundos). Obligando así al driver, que verifique sí los vínculos DDE han recibido, o no, una actualización de los datos durante un periodo específico de tiempo. Un valor de 0 en esta casilla significará que el driver no debe realizar esta acción en ningún momento.

En este último paso de creación del proyecto de servidor DDE Client, se debe crear un tag por cada objeto DDE que estará disponible para cualquier cliente OPC.

Es importante notar que el driver solo aceptara direcciones que tengan una sintaxis válida, pero no comunicará si el servidor DDE específico o el objeto están actualmente disponibles hasta que el "tag" se haya hecho activo por una aplicación cliente. Si el vínculo para este "tag" no se ha podido crear para este momento, la calidad de los datos del "tag" serán considerados como erróneos. Y por consiguiente, un apropiado mensaje de error aparecerá en el registro de eventos del servidor OPC.

Driver Memory Based

- **Descripción**

El driver Memory Based, se añade dentro de un modelo industrial de comunicaciones OPC Server y actúa como un dispositivo de simulación, el cual permite que los usuarios mantengan los valores de "tag" mientras el servidor corre.

- **Configuración**

Tal y como se ha desarrollado con el caso anterior, el primer paso es seleccionar el driver Memory Based de la lista desplegable.

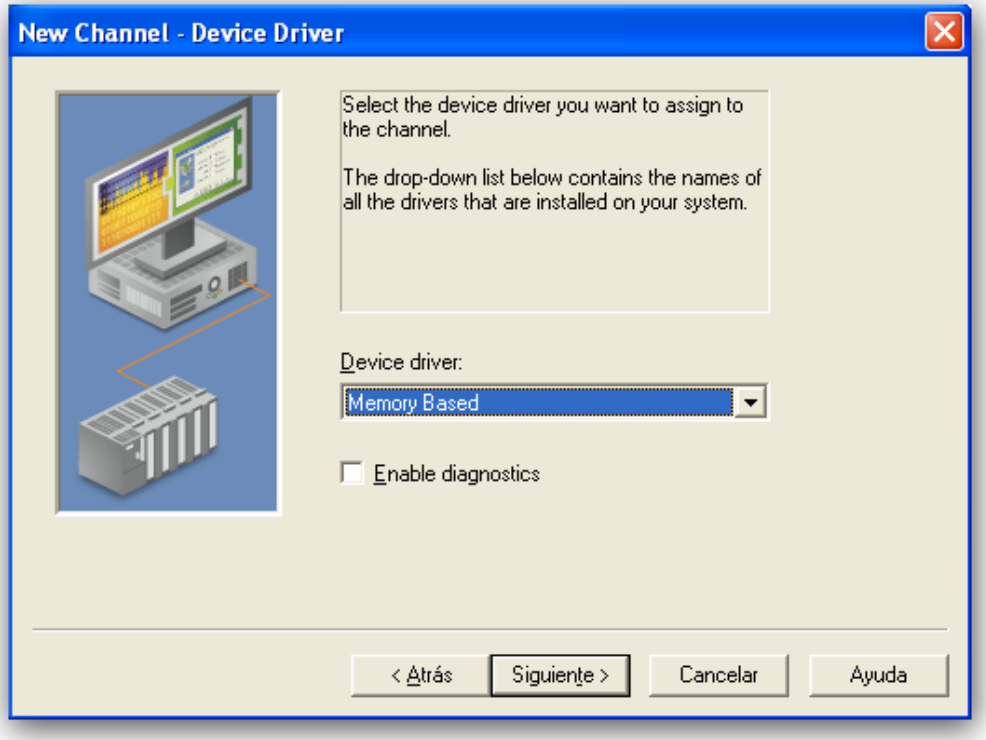

Figura 119. Selección de driver Memory Based

El concepto clave de este driver, se basa en la posibilidad de mantener el valor de los "Tags" mientras el servidor está todavía en marcha. Este valor incluso se guarda después de haber apagado el servidor. Esto se consigue marcando la casilla que se marca en la siguiente figura mientas se está configurando el driver.

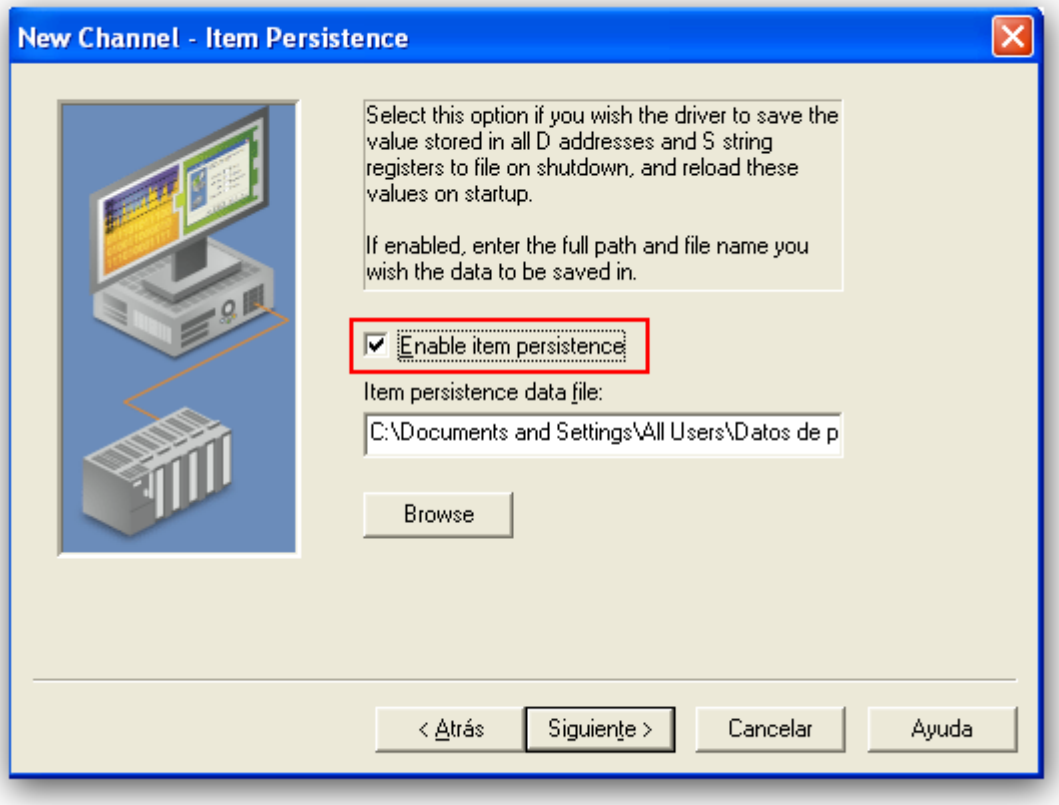

Figura 120. Memory Based driver - Activar la persistencia de objetos

Explicando por encima cada casilla del formulario anterior:

- "*Enable item persistance*" o Permitir la persistencia de objetos: Por defecto esta casilla está inactiva, cuando se activa permite la persistencia de objetos durante el tiempo.

- "*Item persistance data file*" o Archivo de persistencia de objetos: Este parámetro especifica el nombre completo del archivo de datos, al igual que su localización exacta.

Si hay algún problema durante la restauración de datos persistentes cuando el servidor OPC se abre de nuevo. Todos estos datos serán borrados automáticamente.

Driver Advanced Simulator

- **Descripción**

Este será el tipo de driver seleccionado finalmente para la ejecución de este proyecto, debido a su sinergia con bases de datos como SQL y con cualquier tipo de servidor OPC y HMI. Además de ofrecer un vínculo visiblemente más sencillo de entender.

El Advanced Simulator Driver se conecta a cualquier OPC Server dirigido a procesos industriales, que provea acceso a cualquier HMI (Human Machine Interface), SCADA (Supervisory Control And Data Acquisition), Historian o aplicación de empresa que soporte OPC, DDE (Dinamic Data Exchange), FastDDE y SuiteLink.

Advanced Simulator Driver provee, además, un único punto de acceso a múltiples ODBC (Open Database Connectivity). Este simula datos reales leyendo todos los datos guardados en la tabla seleccionada a la velocidad del ciclo de scan que se le imponga.

- **Configuración**

Este tipo de driver soporta cualquier tipo de fuente de datos ODBC, como son:

- Microsoft Access,
- Microsoft SQL,
- MySQL
- Oracle
- Sybase

El protocolo de comunicación utilizado por este driver es ODBC API, y existen ciertos requerimientos que el sistema debe cumplir, como por ejemplo, Los componentes deben ser MDAC (Microsoft Data Access Components), estos consisten en diversos componentes básicos que proveen diferentes tecnologías de bases de datos, incluyendo también ODBC y sus drivers.

A la hora de configurar estos drivers, una de las características importantes a tener en cuenta es la cantidad máxima de canales y dispositivos que pueden crearse dentro de el mismo. Advanced Simulator Driver soporta un número de 100 canales simultáneos, en los cuales pueden ser añadidos 4096 dispositivos.

Se desarrollará la configuración del driver en el apartado de conectividad entre KEPServerEx y MySQL, puesto que el enlace se hace en este punto.

11.4. La aplicación SCADA

La aplicación SCADA realizada, tiene una única función, pese a ser una aplicación muy versátil. Dicha función consiste en representar gráficamente el movimiento de los robots dentro de la nave industrial.

Este apartado se va a dividir en tres sub-apartados. El primer apartado muestra como se ha generado la pantalla visualizador SCADA, en el segundo se hará una demostración del sistema en funcionamiento y las distintas posibilidades de actuación.

11.4.1. Generación de la pantalla SCADA

En primer lugar se generó una nueva imagen de la siguiente forma:

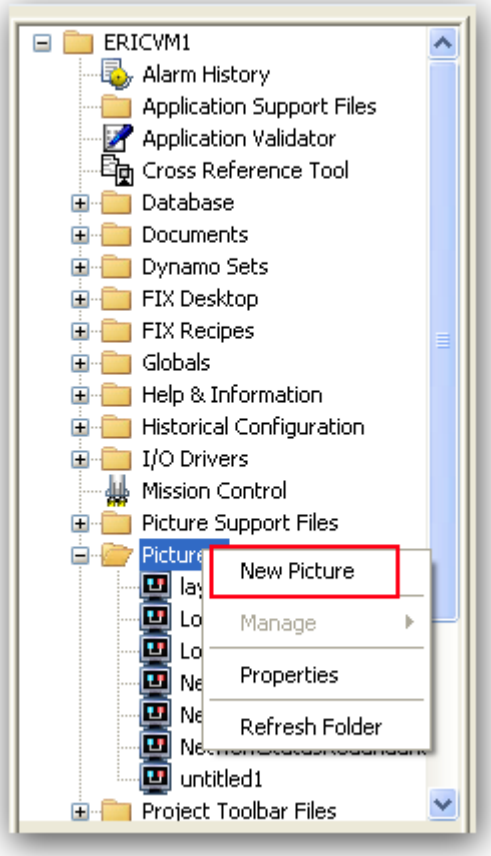

Figura 121. Generación de una nueva imagen en SCADA

Después de generar la figura se insertó una imagen bitmap que contenía la forma del layout. En la siguiente captura de pantalla se muestra como se ejecuta dicha acción.

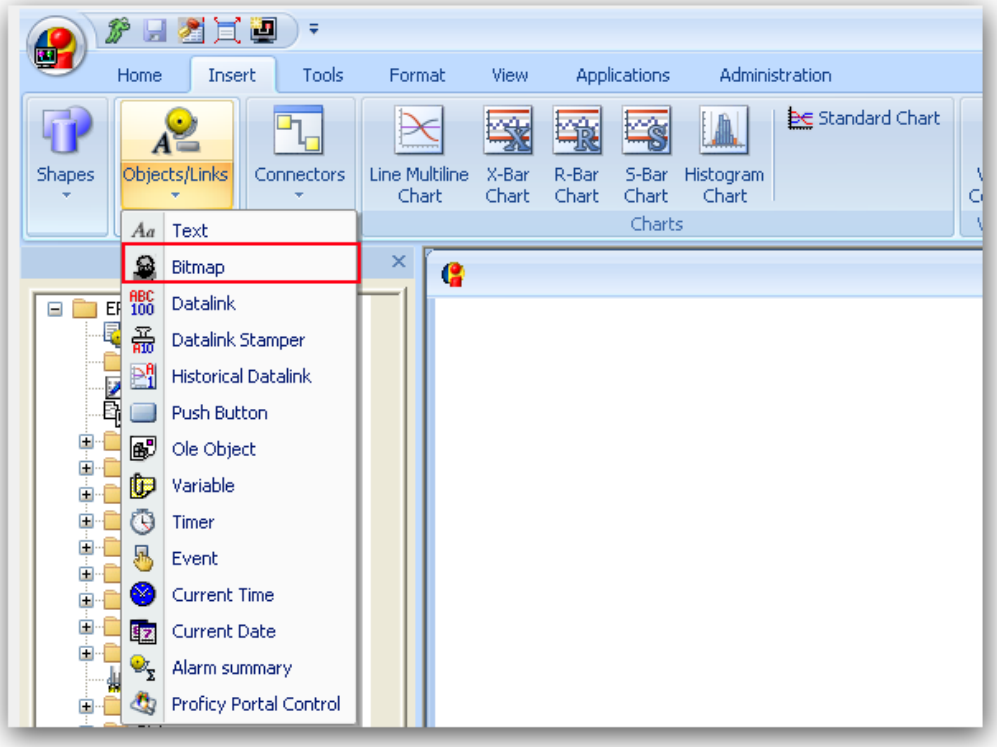

Figura 122. Insertar una imagen en SCADA

El aspecto de la imagen una vez introducida en el programa es el siguiente:

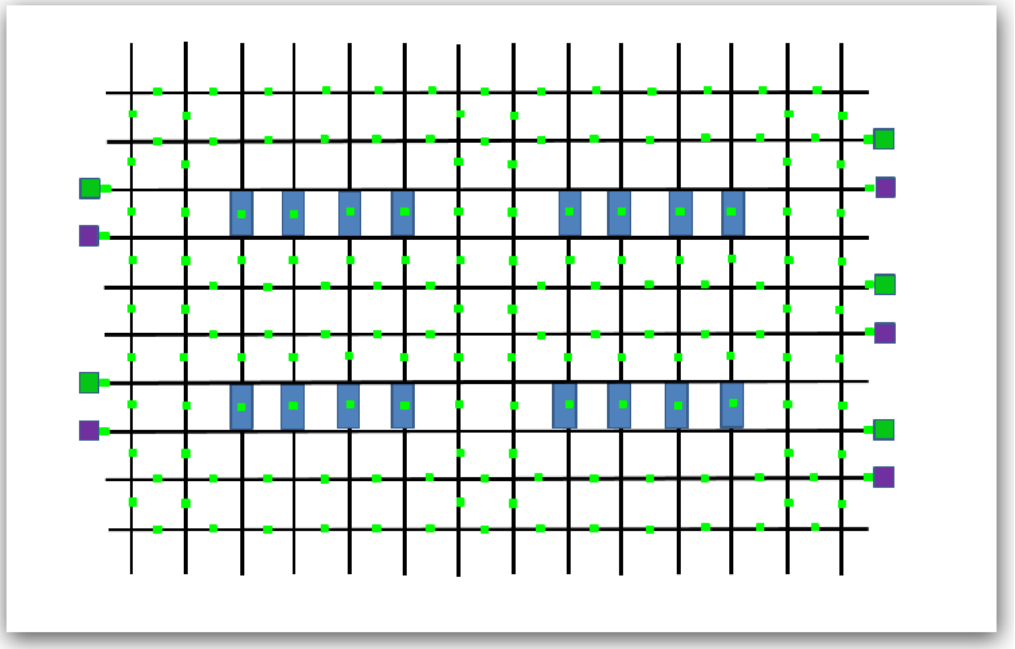

Figura 123. Imagen introducida en SCADA

De esta manera, el layout de la nave industrial predefinida ya está incorporado en la aplicación.

Entonces, se decidió poner objetos con forma cuadrada sobre cada uno de los nodos de la imagen. Un objeto sobre cada nodo por cada uno de los robots Se muestra un ejemplo de esto en la siguiente figura.

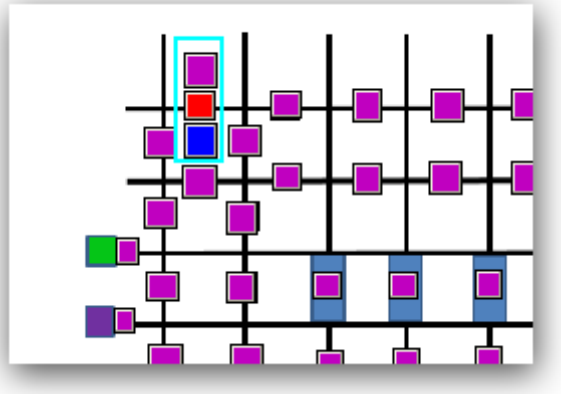

Figura 124. Distribución de objetos en el layout de SCADA

Los tres objetos seleccionados en realidad deben ir superpuestos, pero han sido movidos de lugar para que pueda observarse como se ha realizado realmente este layout. Todos y cada uno de los rest points del layout contiene tres de estos objetos superpuestos, uno por cada robot existente en la nave industrial. Donde los rest points rojos corresponden al robot 1, los azules al robot 2 y los rosas al robot 3.

La distribución definitiva que se observa después de haber introducido todos los objetos es la que se muestra en la figura 125.

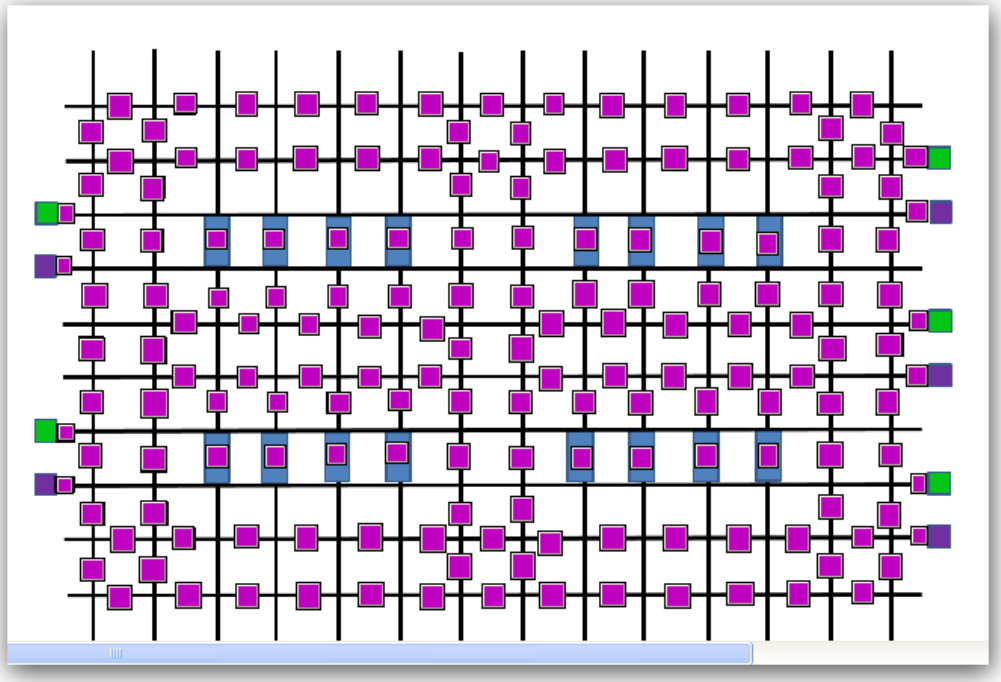

Figura 125. Distribución definitiva de objetos sobre el layout de SCADA

El siguiente paso en el proceso es la conexión de cada uno de los objetos insertados con el cada uno de los tags de los dispositivos del servidor OPC. Sin embargo, este proceso será explicado en la siguiente unidad del proyecto.

11.4.2. Funcionamiento de la aplicación

Para hablar del funcionamiento de la aplicación es necesario volver la vista hacia la aplicación de VisualBasic.net. Al utilizar el botón de enviar pedido en la parte de la aplicación que utiliza el layout predefinido, tal y como se muestra en la figura 126.

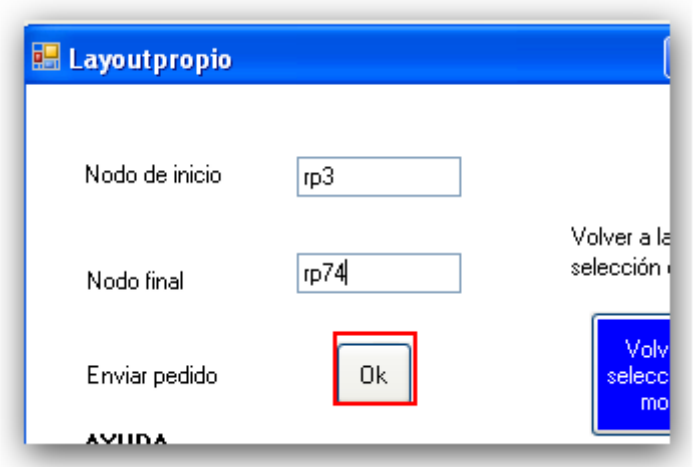

Figura 126. Envío de pedido

Al realizar dicha acción, aparecerá en pantalla el primer robot, transportando el producto desde la posición inicial hasta la posición final tal y como se observa en la siguiente figura. El robot va cambiando de posiciones en función de lo que haya dictaminado el algoritmo Dijkstra.

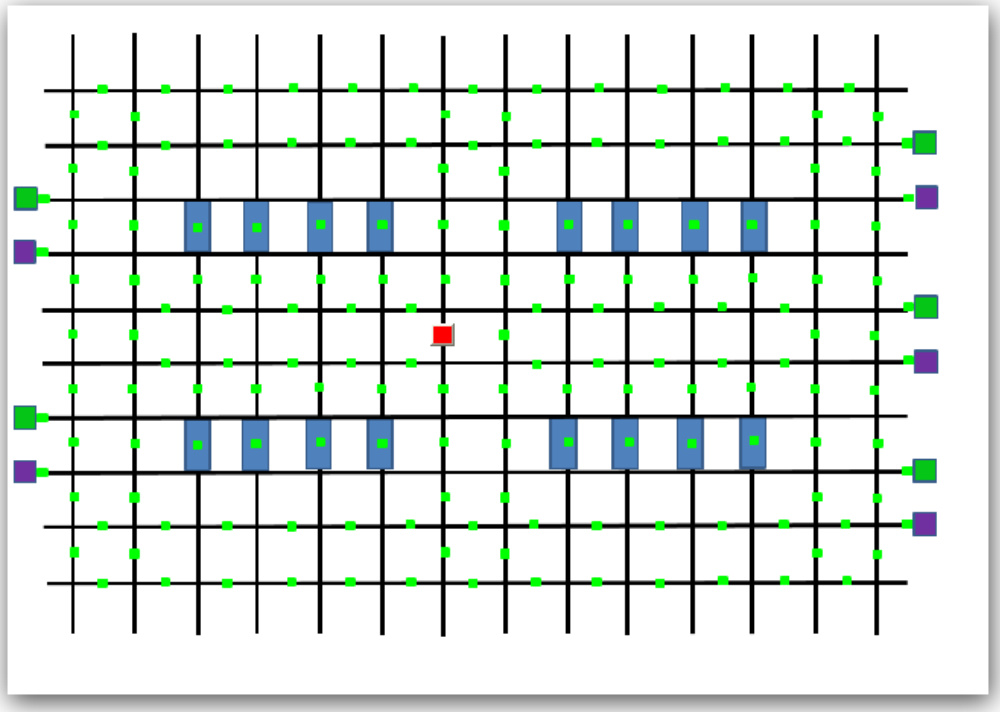

Figura 127. Un robot en funcionamiento

En este momento, el usuario realiza un segundo pedido, haciendo aparecer así un segundo robot realizando la segunda entrega de pedido.

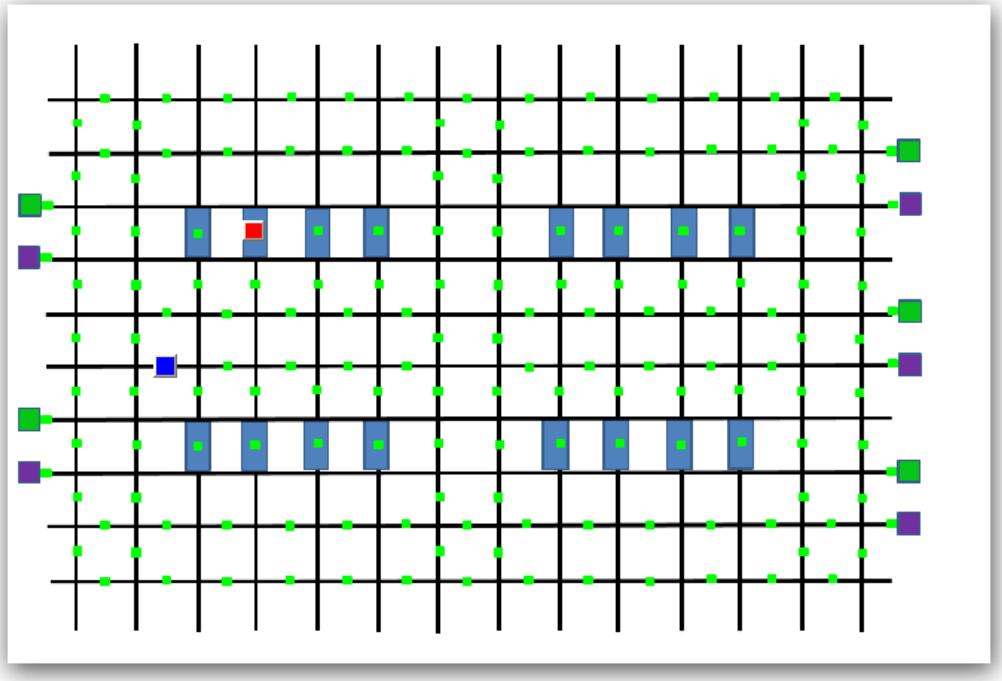

Figura 128. Dos robots funcionando simultáneamente

En el momento en el que el usuario realiza un tercer pedido, siempre y cuando el resto de robots estén ocupados haciendo su tarea, aparecerá el tercer robot y trabajaran los tres de manera simultánea. Se puede observar dicho comportamiento del sistema en la figura 129.

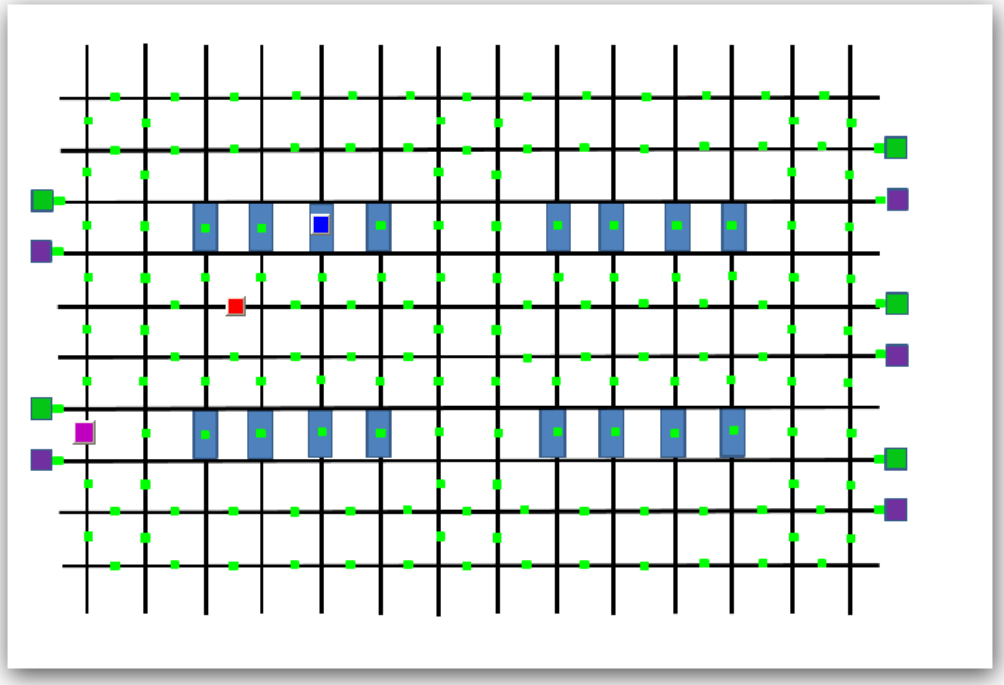

Figura 129. Funcionamiento del sistema con tres robots simultáneos

12. Conexión entre clientes y servidores

En este apartado se mostrará la forma de enlazar los clientes con los servidores que el proyecto incluye, debido a que en la mayor parte de los casos no es un proceso trivial.

12.1. VB.net / SQL Server

La conexión entre la base de datos y la aplicación visual basic.net se ha efectuado dentro del entorno de VB.net.

En primer lugar se debe crear una o diversas tablas en la base de datos SQL tal y como se mostró en la unidad anterior. Posteriormente, se debe añadir un nuevo objeto al proyecto el entorno de VB tal y como se muestra en la figura 130 clicando con botón derecho sobre el nombre del proyecto en la ventana de explorador de solución.

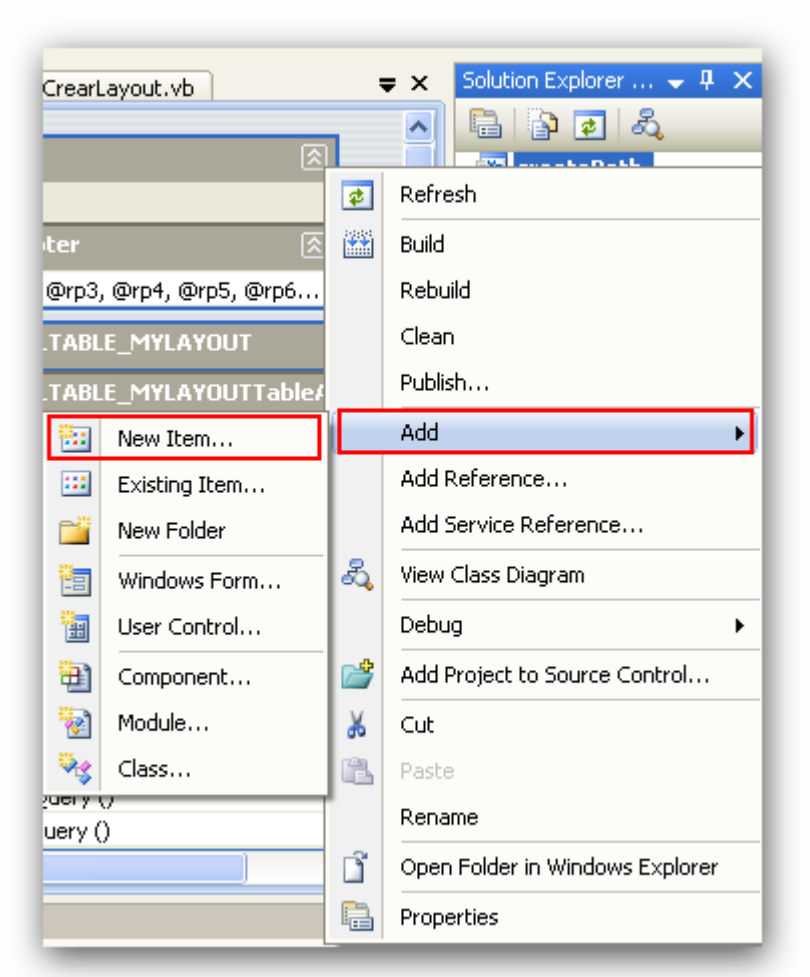

Figura 130. Añadir un nuevo objeto al proyecto

Una vez seleccionado que se quiere añadir un nuevo objeto se abre un nuevo formulario para escoger el tipo de objeto que se desea añadir, para poder enlazar una tabla externa al programa se debe seleccionar el tipo DataSet.xsd y darle un nombre a dicho objeto, clicando posteriormente el botón de añadir (Add).

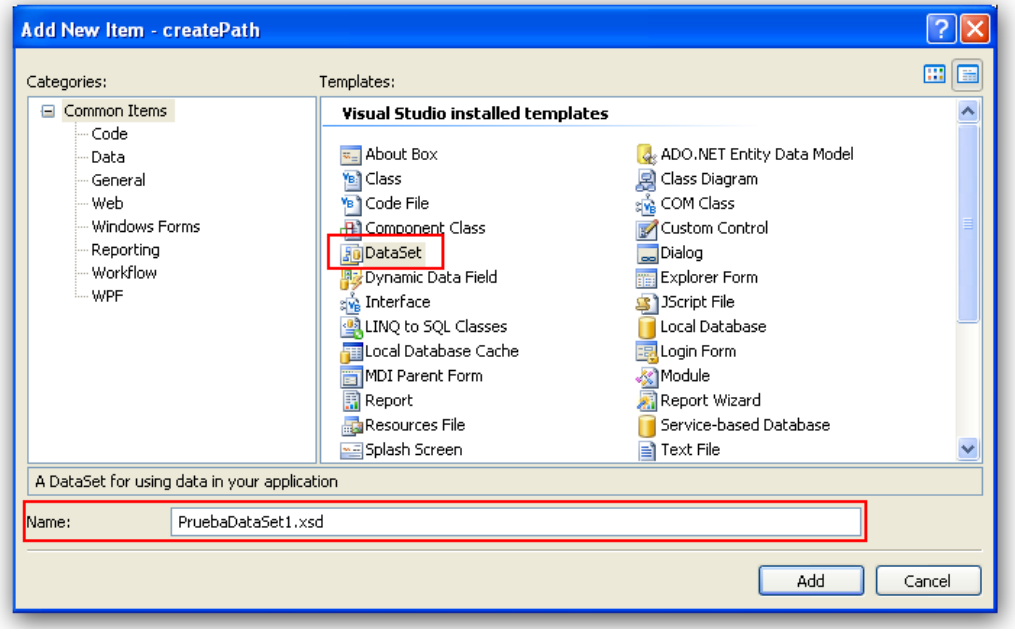

Figura 131. Selección del tipo de objeto DataSet

Al generar el objeto este aparecerá en el explorador de proyecto, y el siguiente paso a seguir será abrir la una ToolBox específica para la introducción de objetos relacionados con tablas y bases de datos. Dentro de dicha ToolBox puede encontrarse el objeto "TableAdapter" que se debe arrastrar hacia la pestaña el nuevo objeto introducido tal y como se observa en la próxima imagen.

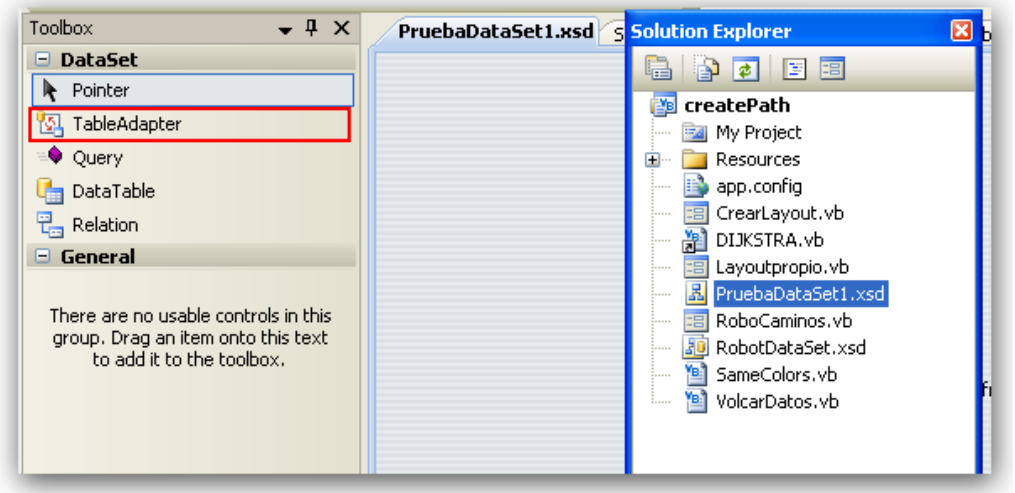

Figura 132. Inserción de un adaptador de tabla al objeto de prueba creado

En el momento en el que agregas el objeto TableAdapter el sistema pide que se le agregue el tipo de base de datos y la base de datos en cuestión desde la cual se quiere extraer la información, tal y como se muestra en la figura 133.

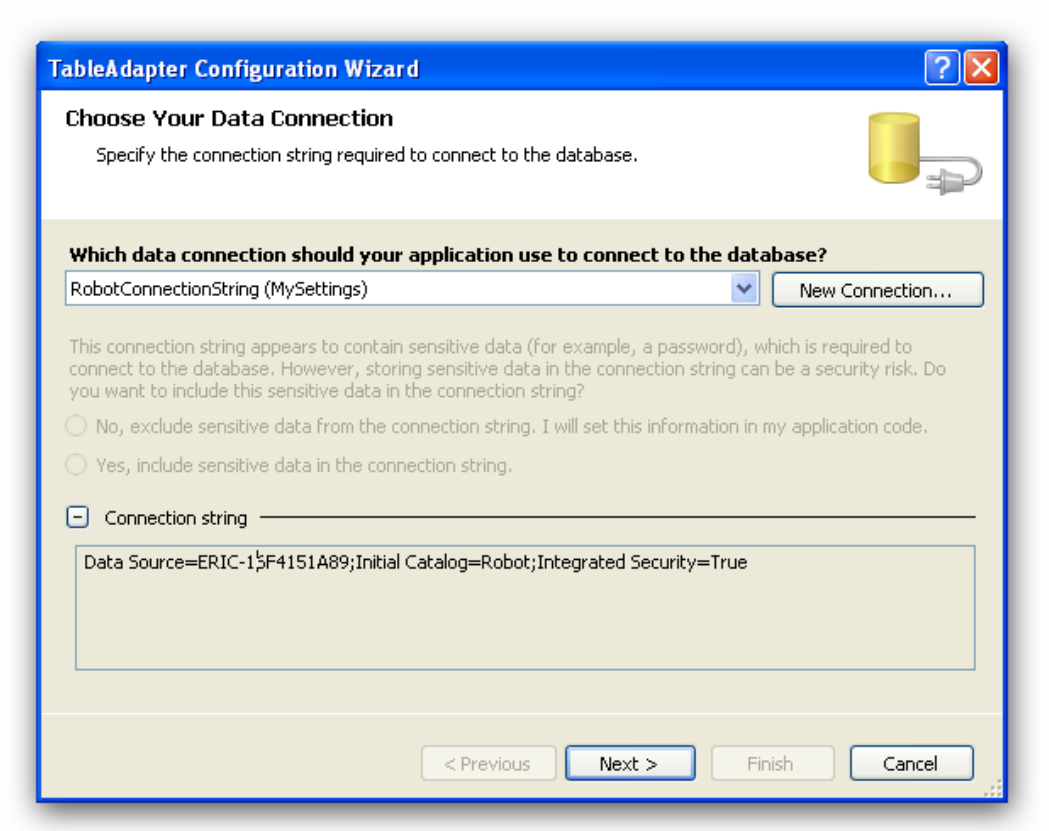

Figura 133. Configuración de la base de datos

Al pulsar siguiente después de seleccionar la base de datos en la cual se han creado las tablas que se quiere enlazar, el sistema pedirá al usuario que se le introduzca el código SQL, mediante el cual, se extraerá la información de la tabla perteneciente a la base de datos enlazada. Una vez hecho esto, con todas las tablas necesarias se pasará a introducir las sentencias necesarias de SQL que se vayan a utilizar. En el proyecto que se ha llevado a cabo, se han utilizado sentencias SELECT, que sirven para extraer valores de una tabla en cuestión. Se han utilizado sentencias INSERT, para introducir valores en dichas tablas. También sentencias DELETE para refrescar los valores de las tablas que ya quedaron obsoletos, y finalmente también se han añadido la llamada de procesos almacenados. Este último es un paso crítico para el funcionamiento del sistema, puesto que tal y como se comentó en unidades anteriores, de esta forma se consiguen dos objetivos distintos. El primero, reducir la cantidad de código utilizado y el segundo y dependiente del primero, se mejora el rendimiento de la aplicación en gran medida. Una vez todas esas sentencias propias de SQL han sido insertadas en el sistema, estas pueden pasar a

utilizarse como si se tratase con tablas pertenecientes a la aplicación de VB.net. Se muestra en la siguiente captura de pantalla como ha quedado el sistema después de la inserción de todas las tablas y sus procesos.

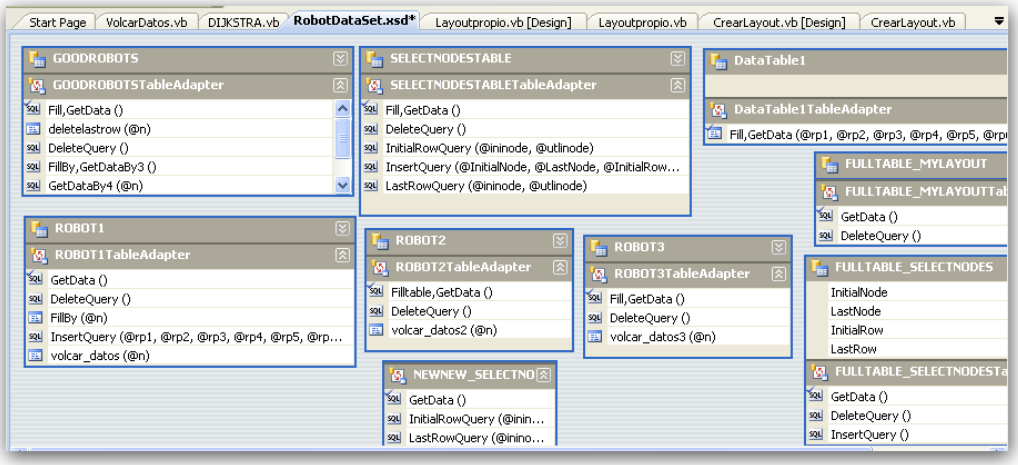

Figura 134. Estructura de tablas insertadas en VB.net desde SQL

12.2. SQL Server / KEPServerEX 5.0

Después de enlazar la base de datos SQL Server con la aplicación VB.net, se procede a enlazar las tablas creadas en SQL Server con el servidor OPC. Puesto que ha sido por este motivo que se ha decidido utilizar tablas SQL dentro del diagrama de flujo del programa.

El paso previo a realizar dicha conexión, consiste en realizar alguna modificación en el sistema de Windows para que el servidor OPC pueda visualizar la base de datos. Primeramente se debe acceder a la carpeta "Herramientas Administrativas" dentro del panel de control. Dentro de dicha carpeta, se accederá a la aplicación que se muestra a continuación.

Figura 135. Acceso a Origenes de datos (ODBC)

Una vez dentro del administrador de orígenes de datos (ODBC) se debe proceder a agregar un nuevo origen de tipo SQL Server utilizando los datos de acceso de la base de datos creada en SQL. Tal y como se muestra en la próxima figura.

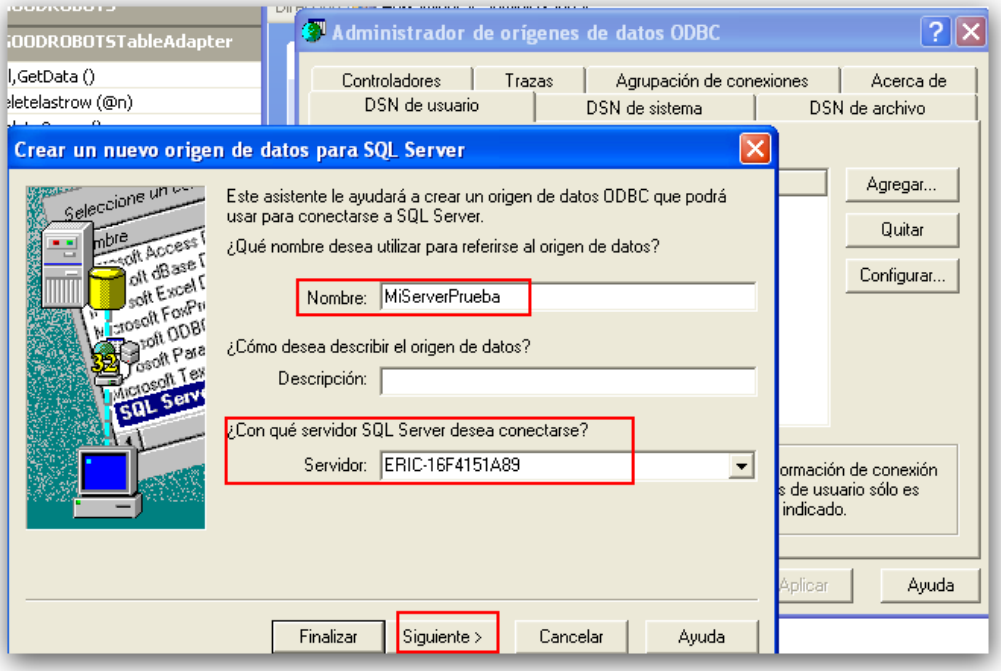

Figura 136. Creación de un nuevo origen de datos SQL Server

Una vez añadido dicho origen de datos, aparecerá el nuevo origen de datos para poderlo enlazar con el servidor OPC. Se puede observar dicho origen de datos en la próxima imagen.

Figura 137. Proceso de creación de origen finalizado

Una vez este paso ha sido finalizado, ya puede accederse al servidor OPC KEPServerEX y añadir enlazar el servidor con cualquiera de las tablas creadas dentro de la base de datos SQL. Se puede observar en la figura 138 el método para realizar el enlace dentro de dicho servidor.

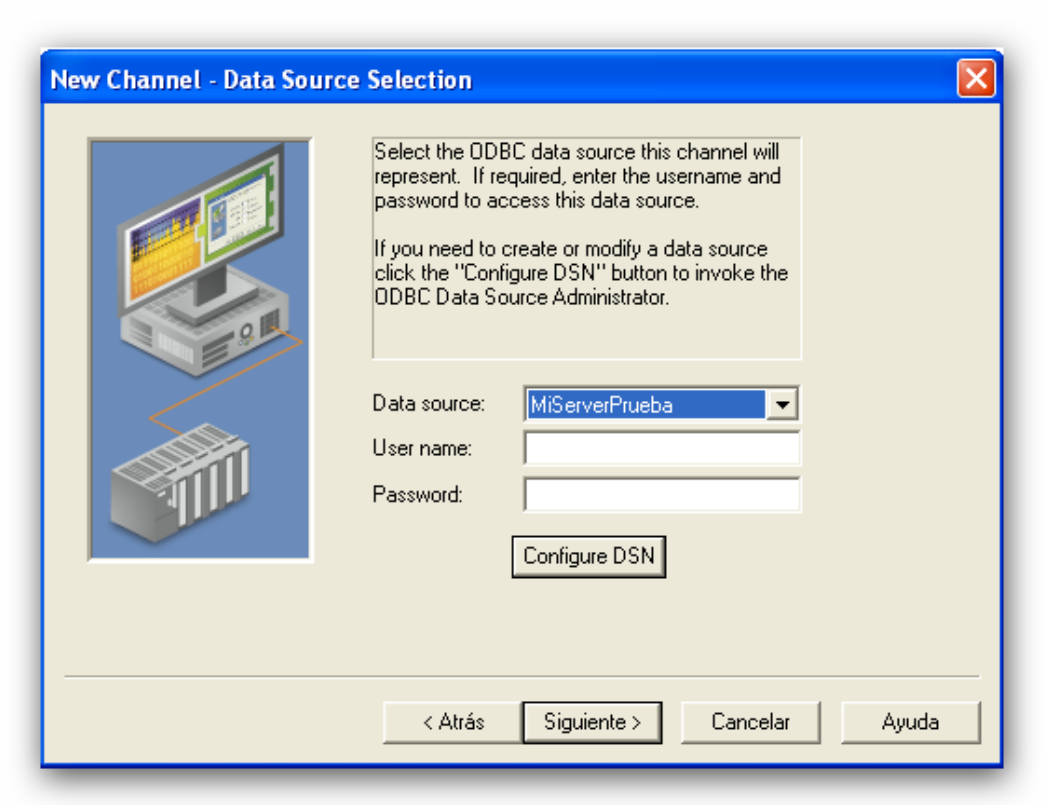

Figura 138. Selección de la fuente de datos

A la hora de configurar un nuevo canal, después de haber seleccionado el driver Advanced Simulator, el sistema requiere introducir la fuente de datos desde la cual se realizará el traspaso de datos. Abriendo el desplegable, se encontrará la opción recientemente introducida en el sistema.

Una vez se haya creado el canal y se disponga a generar un dispositivo, uno de los pasos para generar dicho dispositivo será realizar la elección de una tabla en cuestión. En el caso que comprende este proyecto, las tablas ROBOT1, 2 y 3. Puede observarse en la próxima figura que dichas tablas aparecen como opción tras haber generado el origen de datos adecuadamente.

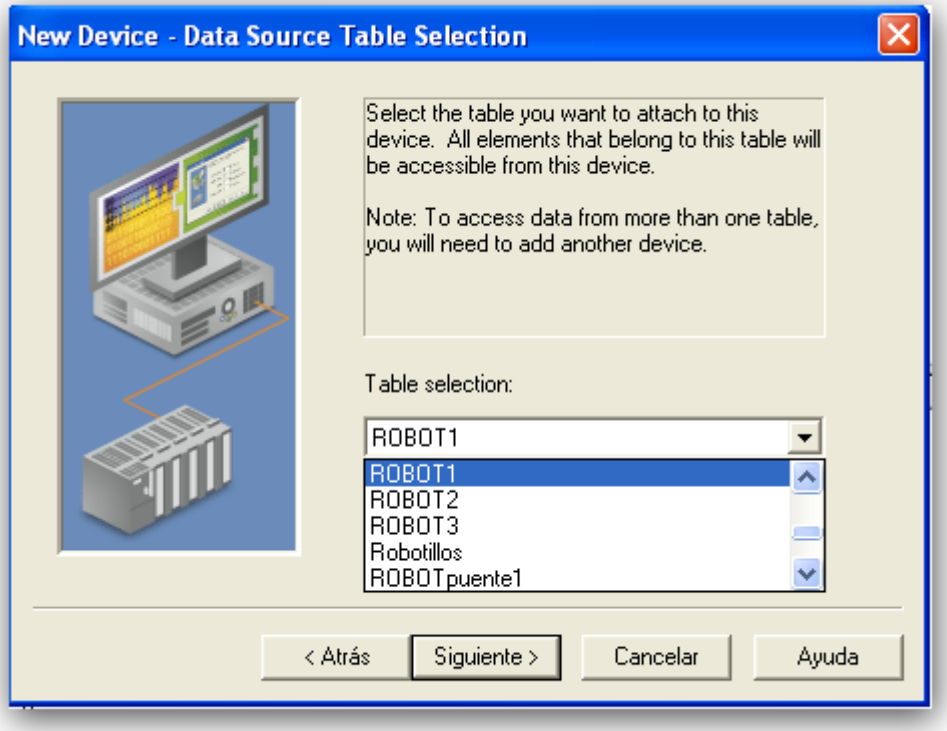

Figura 139. Selección de tabla para cada dispositivo

Una vez la tabla ha sido seleccionada podrá observarse como se generan los tags correspondientes a cada una de las columnas de forma automática.

Tag Name	Address	Data Type	Scan Rate	Scaling	Description
MoveTo	<INTERNA...	Long	100	None	
ROBOT1_Fila	ROBOT1.Fila	Long	100	None	
ROBOT1_rp1	ROBOT1.rp1	Long	100	None	
ROBOT1_rp10	ROBOT1.rp10	Long	100	None	
ROBOT1_rp100	ROBOT1.rp...	Long	100	None	
ROBOT1_rp101	ROBOT1.rp...	Long	100	None	
ROBOT1_rp102	ROBOT1.rp...	Long	100	None	
ROBOT1_rp103	ROBOT1.rp...	Long	100	None	
ROBOT1_rp104	ROBOT1.rp...	Long	100	None	
ROBOT1_rp105	ROBOT1.rp...	Long	100	None	
ROBOT1_rp106	ROBOT1.rp...	Long	100	None	
ROBOT1_rp107	ROBOT1.rp...	Long	100	None	
ROBOT1_rp108	ROBOT1.rp...	Long	100	None	

Figura 140. Generación de tags de forma automática

12.3. VB.net / KEPServerEx 5.0

El objetivo de la conexión entre VB.net y el servidor OPC KEPServerEx 5.0 es únicamente para limpiar las tablas pertenecientes a cada uno de los robots.

Esto se lleva a cabo mediante el cliente OPC ClientACE del mismo desarrollador que el Servidor OPC KEPServerEx 5.0. El requisito previo para poder realizar dicha conexión es instalar en la computadora el cliente OPC ClientACE antes de instalar VB.net. Si la instalación de programas se realiza en este orden, aparecerá una nueva ToolBox en VB.net que permitirá enlazar unos objetos llamados DA Junction a objetos pertenecientes al formulario creado con VB.net.

Primeramente se arrastra sobre el formulario el objeto DA Junction, tal y como se muestra en la siguiente figura.

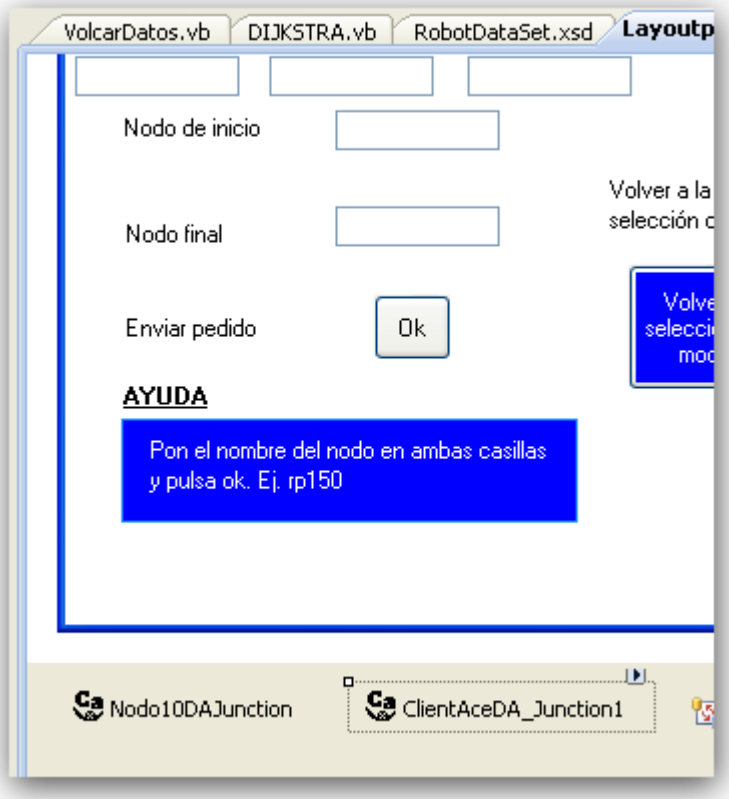

Figura 141. Inserción de DA Junction en el formulario de VB

Una vez se ha arrastrado el objeto al formulario mediante el cual se debe realizar el enlace, se debe visualizar las propiedades de dicha DA Junction para poder entrar en su configuración. Se muestra en la figura 142 el proceso a seguir para acceder a la configuración de la unión entre VB.net y KEPServerEx.

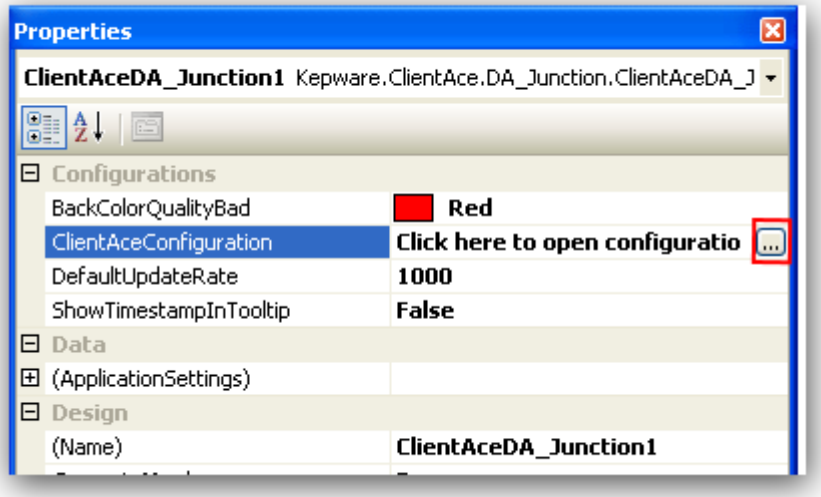

Figura 142. Acceso a la configuración de ClientAce

Accediendo mediante el botón marcado en la captura de pantalla anterior, se accederá al formulario para realizar el enlace. En la siguiente figura puede observarse dicho formulario.

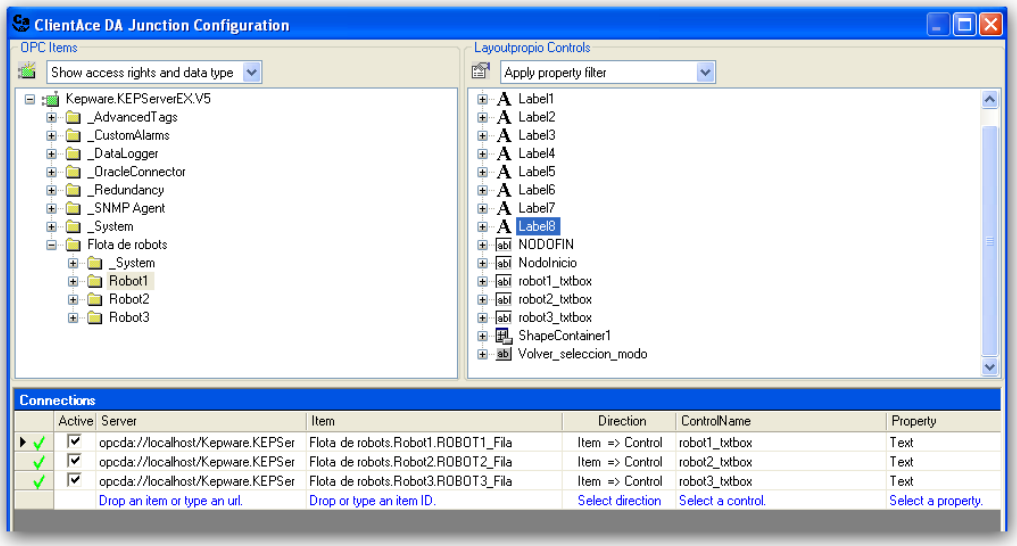

Figura 143. Configuración de ClientACE DA Junction

En la parte izquierda se observan todos los tags que pueden extraerse del servidor de KEPWARE, mientras que en la parte derecha se encuentran los objetos del formulario de Visual Basic.net, se debe arrastrar tanto el objeto OPC como el objeto que se desea controlar del formulario de VB. Una vez hecho esto, puede monitorizarse desde VB.net el valor de cualquiera de los tags pertenecientes a cualquiera de los dispositivos del servidor OPC, utilizándolos a conveniencia.

12.4. KEPServerEx 5.0 / SCADA de iFix

Este enlace corresponde al último enlace de la cadena, el enlace que se realiza para incorporar el sistema de visualización al proyecto. Se desarrolla desde el interior de SCADA.

Existen dos formas de realizar dicho enlace, mediante la base de datos de iFix utilizando el archivo de servicio OPC de iFix , o bien mediante acceso remoto a otros servidores OPC. En este proyecto se ha realizado el segundo tipo de conexión por dos motivos. Por simplicidad de realización y por mejora en el rendimiento, ya que al no tener que acceder a la base de datos de SCADA la velocidad de iteración es mayor al hacer pasar la información a través de menos filtros.

Se ha enlazado cada uno de los objetos añadidos a la pantalla de SCADA con uno de los tags de los dispositivos del servidor OPC. Se puede observar cómo se ha realizado dicho enlace en las siguientes ilustraciones.

Pulsando el botón derecho del ratón sobre cualquier objeto de la pantalla de SCADA, observaremos la opción animación tal y como puede observarse en la figura 144.

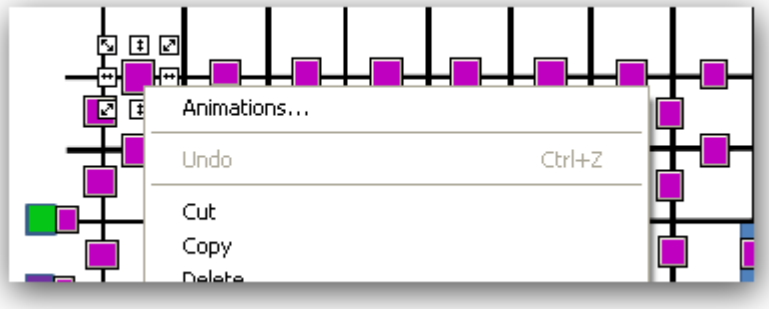

Figura 144. Animación de objetos en SCADA

Al realizar la animación del objeto aparecerá la opción visualización, que será la seleccionada para la realización de este proyecto, se puede visualizar dicho proceso en la figura 145. Al pulsar sobre la opción de visualización aparecerá por pantalla las opciones de configuración posibles tal y como se puede observar en la figura 146.

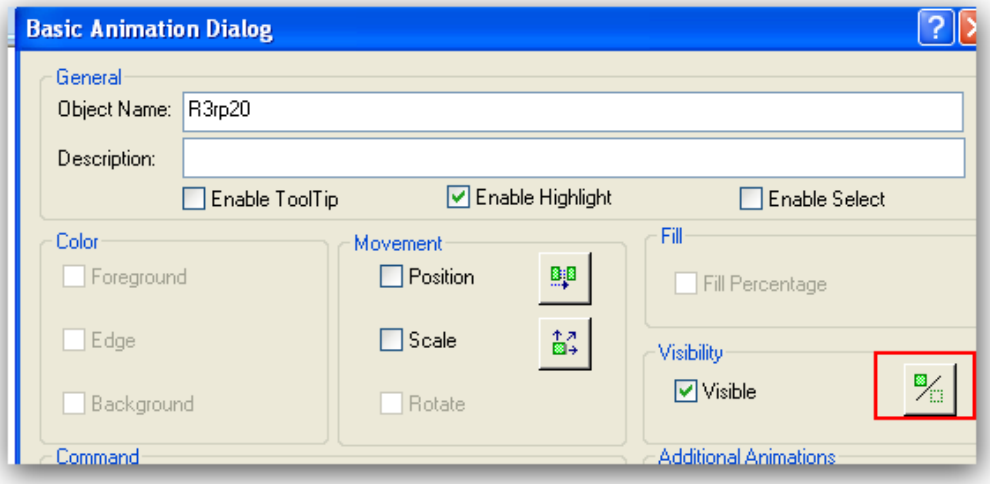

Figura 145. Opciones de animación para los objetos de SCADA

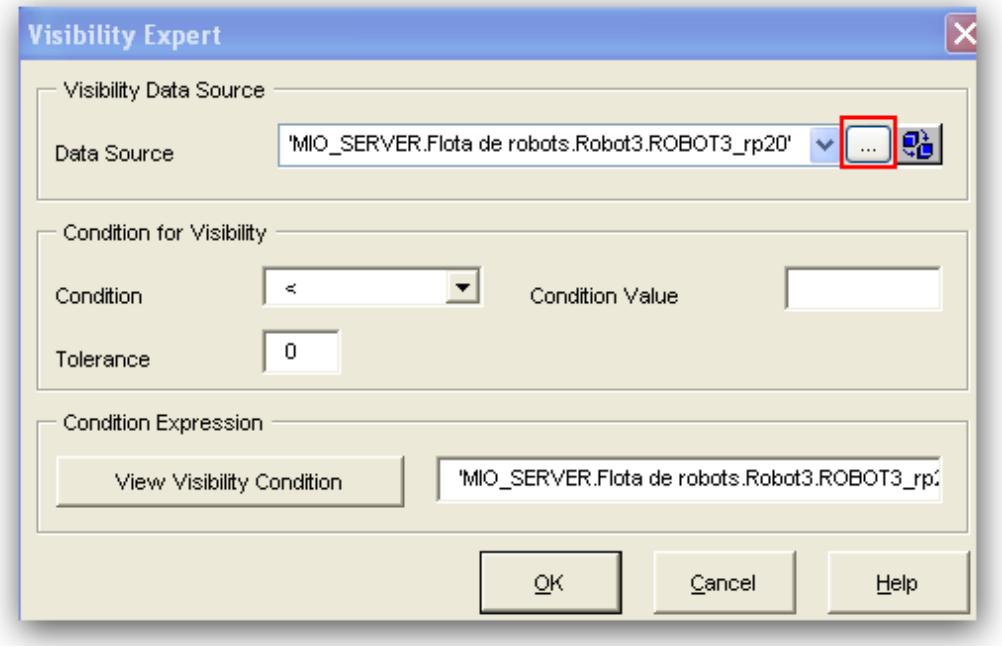

Figura 146. Configuración de visibilidad de objetos

Una vez se ha seleccionado explorar para visualizar los tipos de fuentes de datos que se pueden seleccionar, aparecerá la siguiente cabecera de formulario.

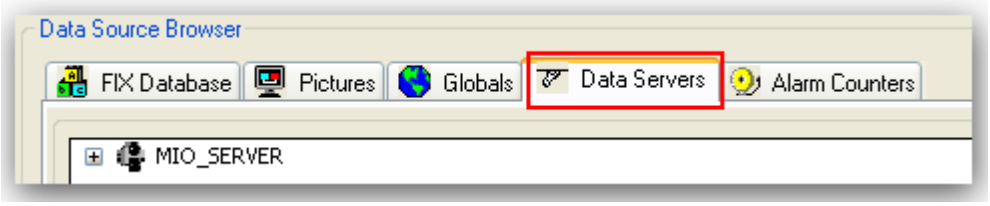

Figura 147. Selección de fuentes de datos

Dentro de dicho formulario se pueden observar diversas pestañas, la que se utilizará para la realización de este proyecto es la que está seleccionada en rojo en la imagen previa. Dentro de los Servidores de datos podremos encontrar cualquier servidor OPC que haya sido creado en la computadora sobre la que se está trabajando. En este caso únicamente se puede encontrar el servidor 'MIO_SERVER' que es donde se pueden encontrar los tags disponibles en los dispositivos de KEPServerEX. Se puede observar lo anteriormente comentado en la figura 148.

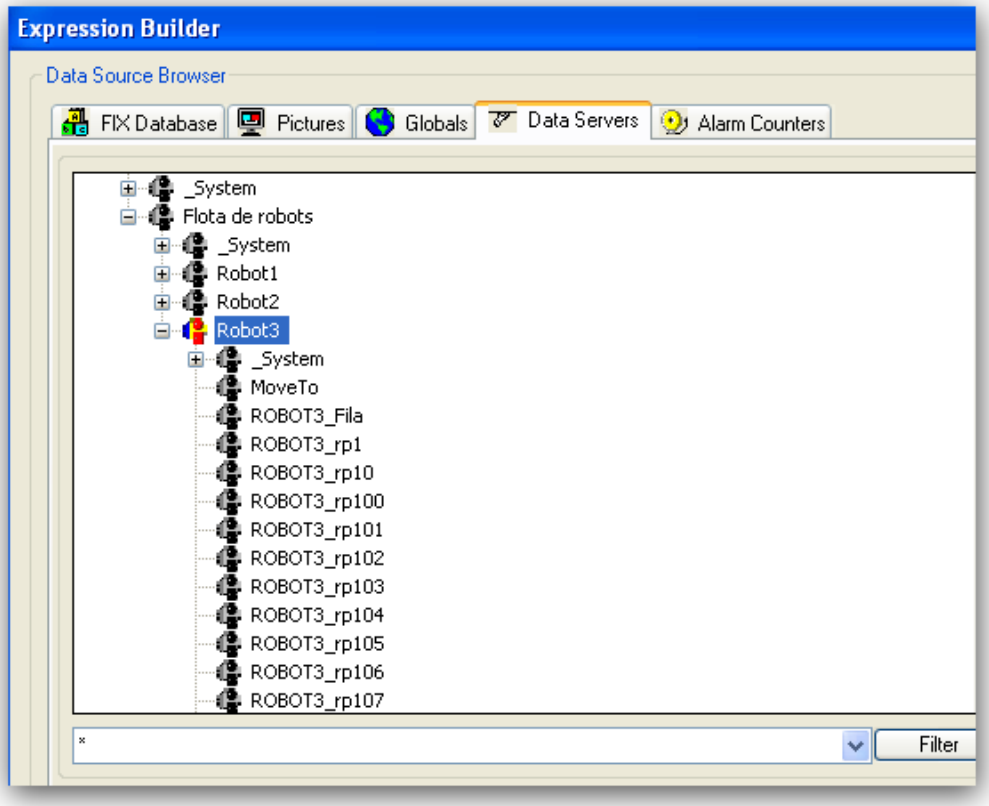

Figura 148.Servidor de datos Flota de robots dentro del configuración de animación

Este proceso debería realizarse tantas veces como objetos haya en el layout representado en SCADA, es decir 276*3 = 826 veces.

Parte IV

RESULTADOS, CONCLUSIONES Y PROPUESTAS DE MEJORA

13. Ejemplo de utilización de la aplicación

En este apartado se realizará un ejemplo práctico del funcionamiento de la aplicación en una situación normal de trabajo.

En primer lugar se accederá a la parte de la aplicación que utiliza el layout predeterminado, y se procederá a realizar el envío de un pedido. Este se trata de una entrada de producto por el nodo "rp3" y se debe llevar al almacén que se encuentra en el nodo "rp165" tal y como se muestra en la siguiente figura:

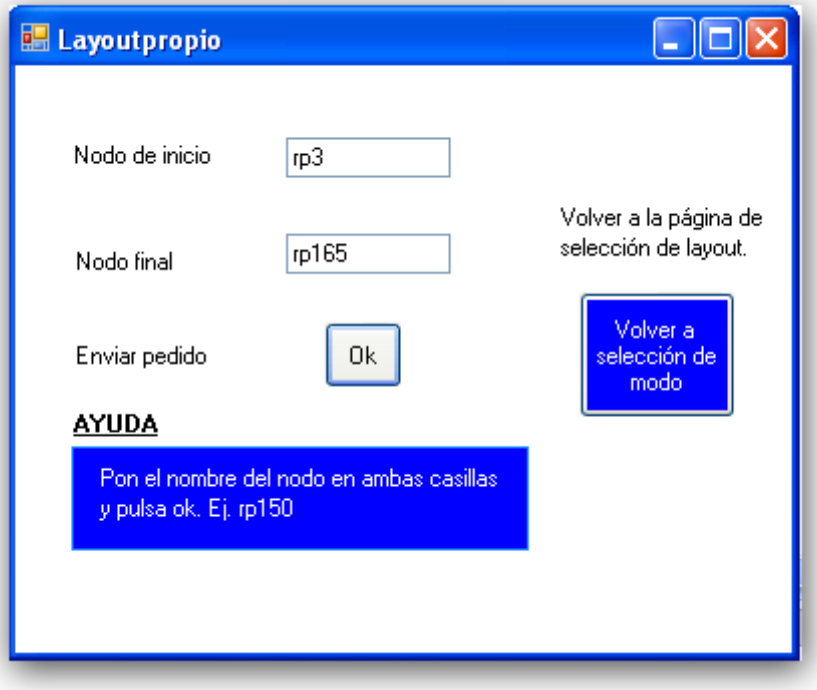

Figura 149. Introducción del primer pedido

En este momento, el sistema comprueba si el robot 1 se encuentra libre, al estar libre, la tarea se le asigna a él directamente. De mientras, en la aplicación SCADA se puede ir observando la realización de dicha tarea por el robot 1. A continuación se puede visualizar todos y cada uno de los pasos realizados por el robot. (Se debe tener en cuenta que los robots se moverán en función de los caminos que se muestran en la figura 60 dentro de la unidad 10 de este proyecto.

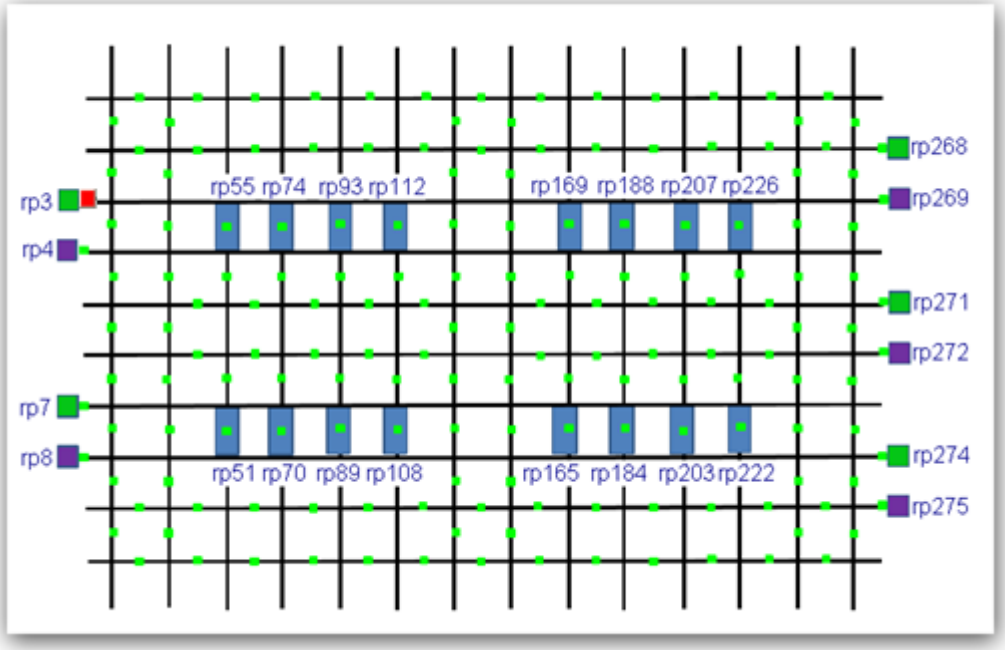

Figura 150. Ejemplo de utilización de la aplicación(1)

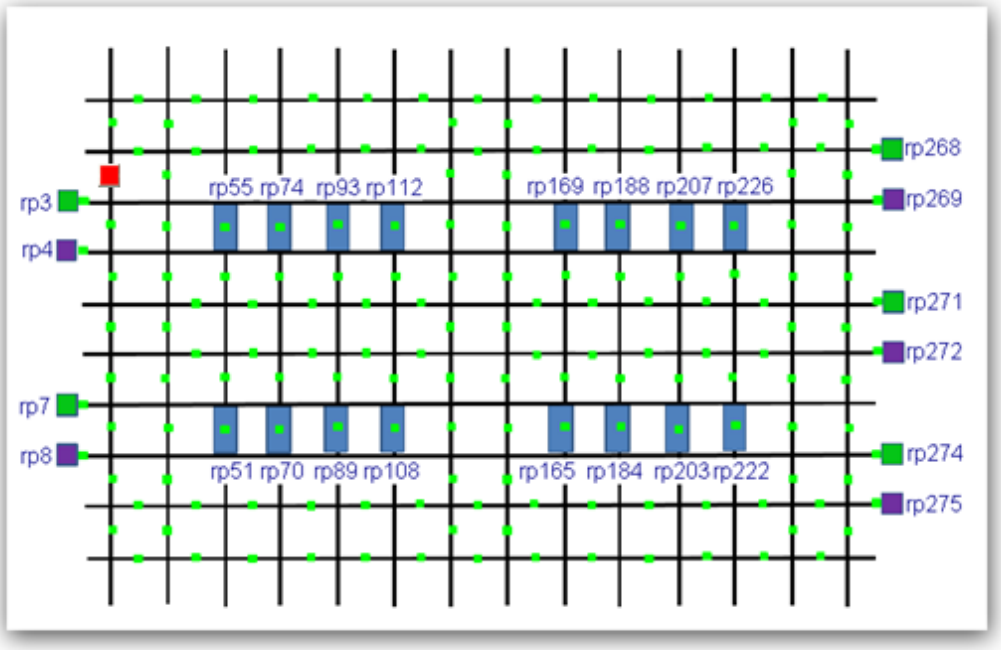

Figura 151. Ejemplo de utilización de la aplicación(2)

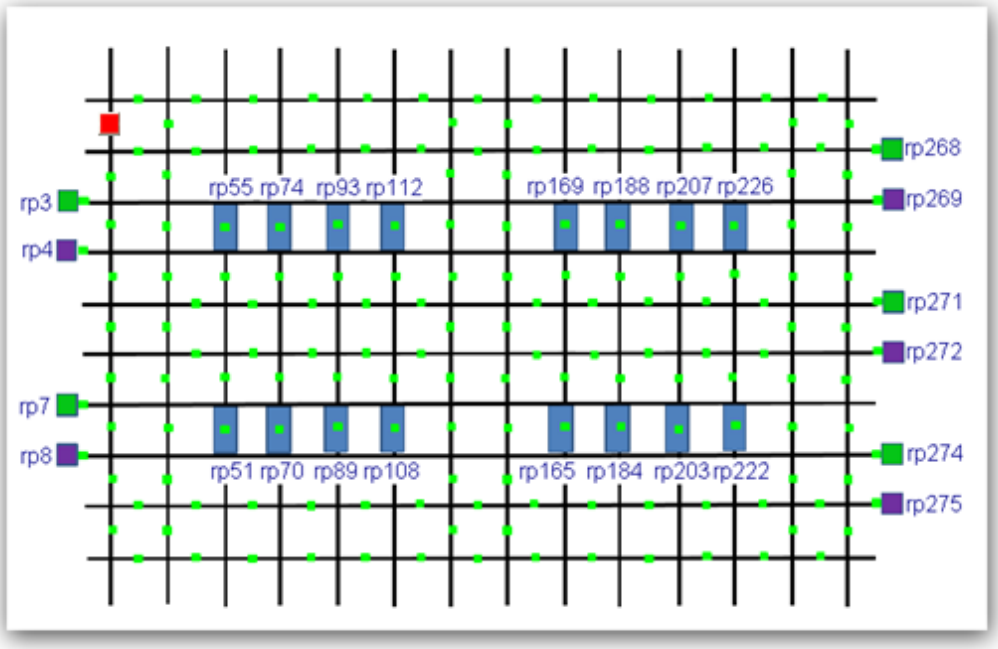

Figura 152. Ejemplo de utilización de la aplicación(3)

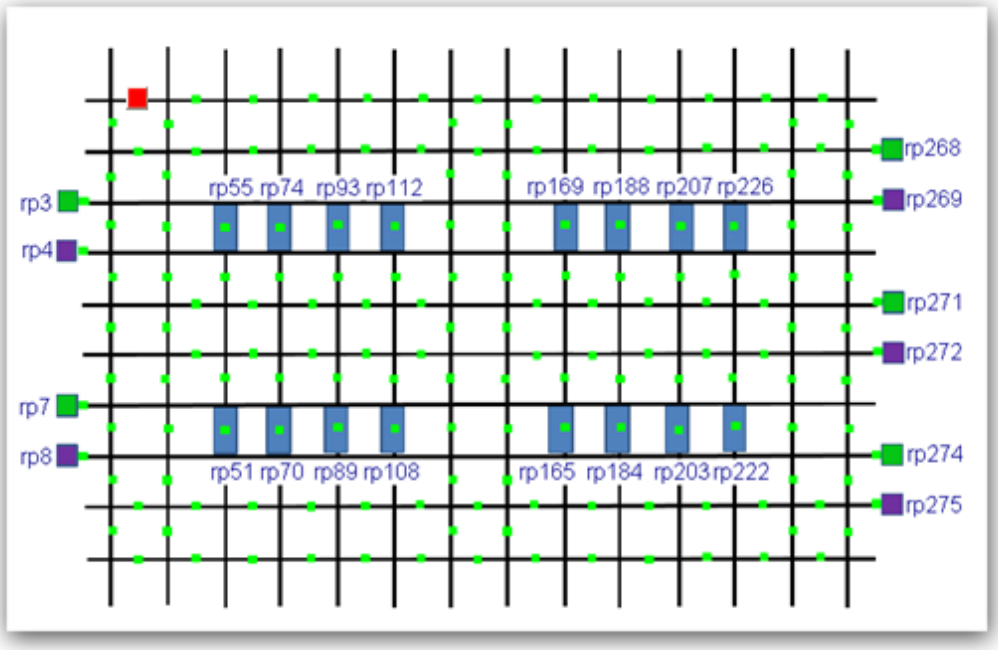

Figura 153. Ejemplo de utilización de la aplicación(4)

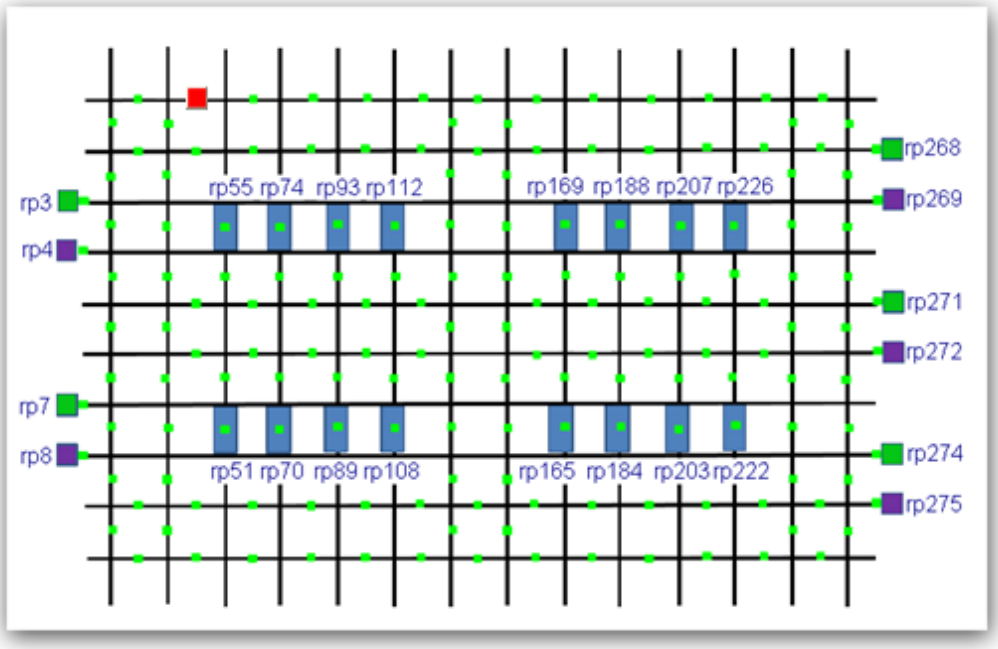

Figura 154. Ejemplo de utilización de la aplicación(5)

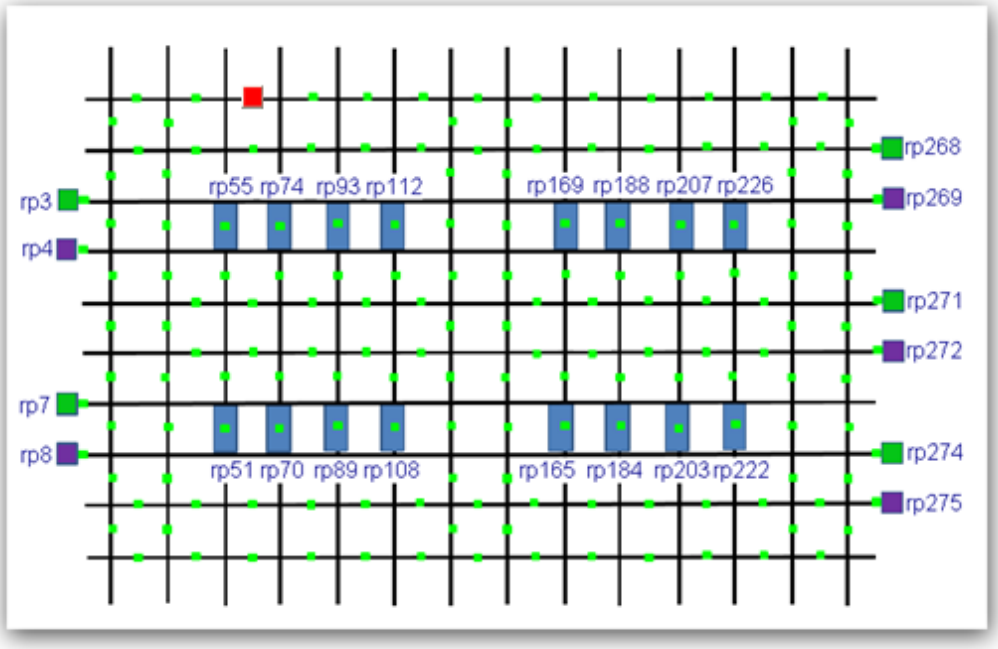

Figura 155. Ejemplo de utilización de la aplicación(6)

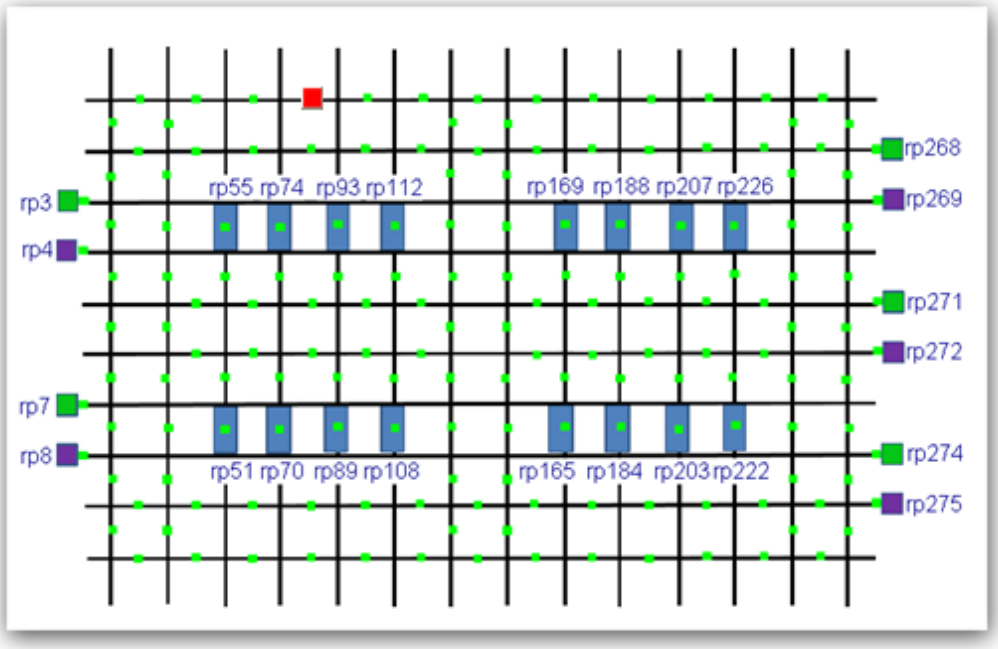

Figura 156. Ejemplo de utilización de la aplicación(7)

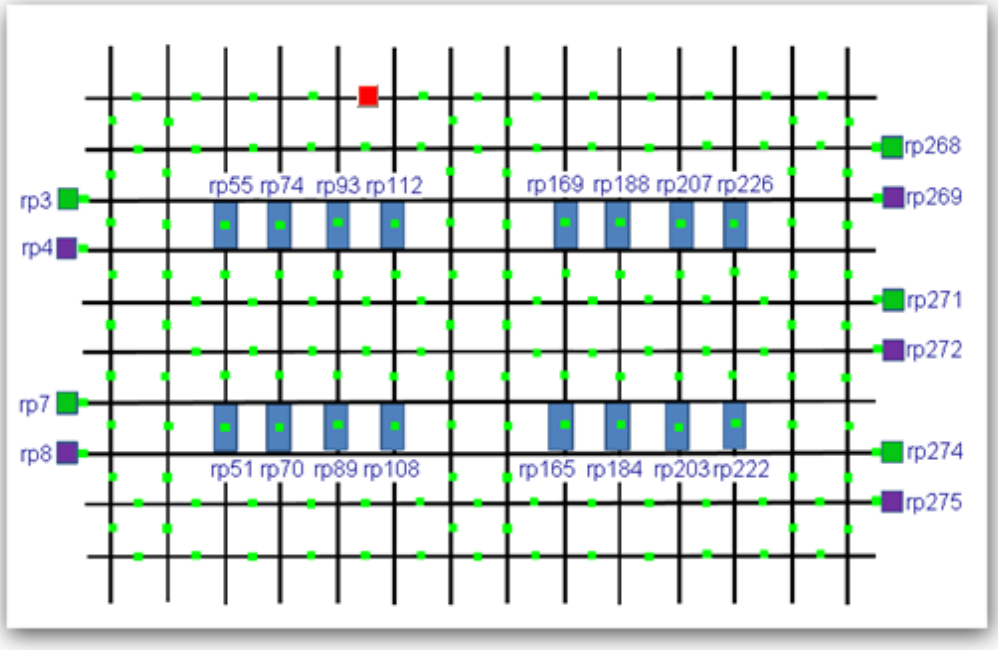

Figura 157. Ejemplo de utilización de la aplicación(8)

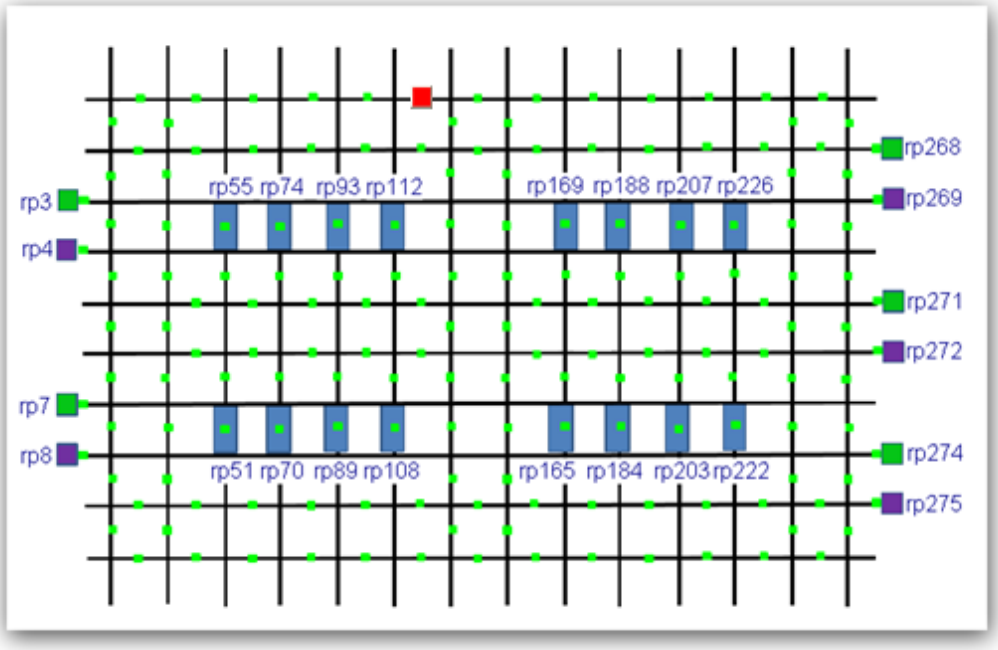

Figura 158. Ejemplo de utilización de la aplicación(9)

En este momento se realiza un segundo pedido, el sistema chequea si el robot 1 se encuentra disponible y al no estarlo pasa a comprobar si el robot 2 se encuentra libre. Al encontrarse libre de carga, el robot 2 pasa a ocuparse de la tarea. En la próxima figura puede observarse el segundo pedido enviado por el usuario.

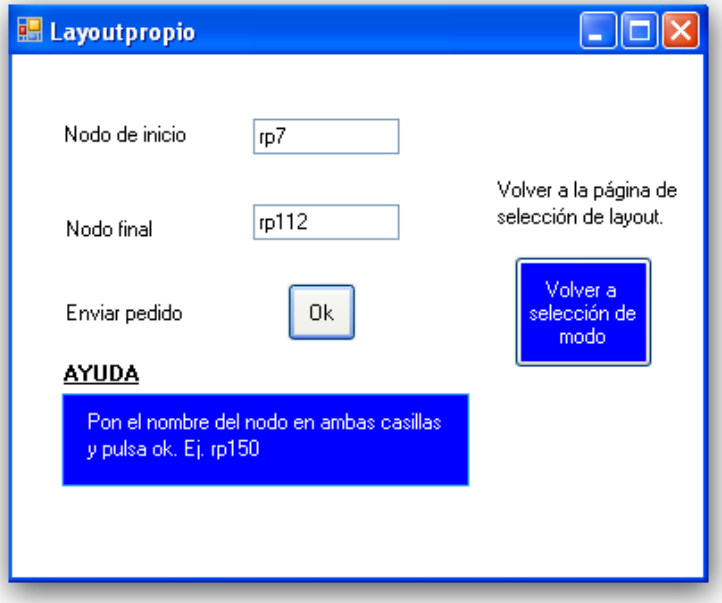

Figura 159. Introducción del segundo pedido

Así pues, también entra en acción el segundo robot.

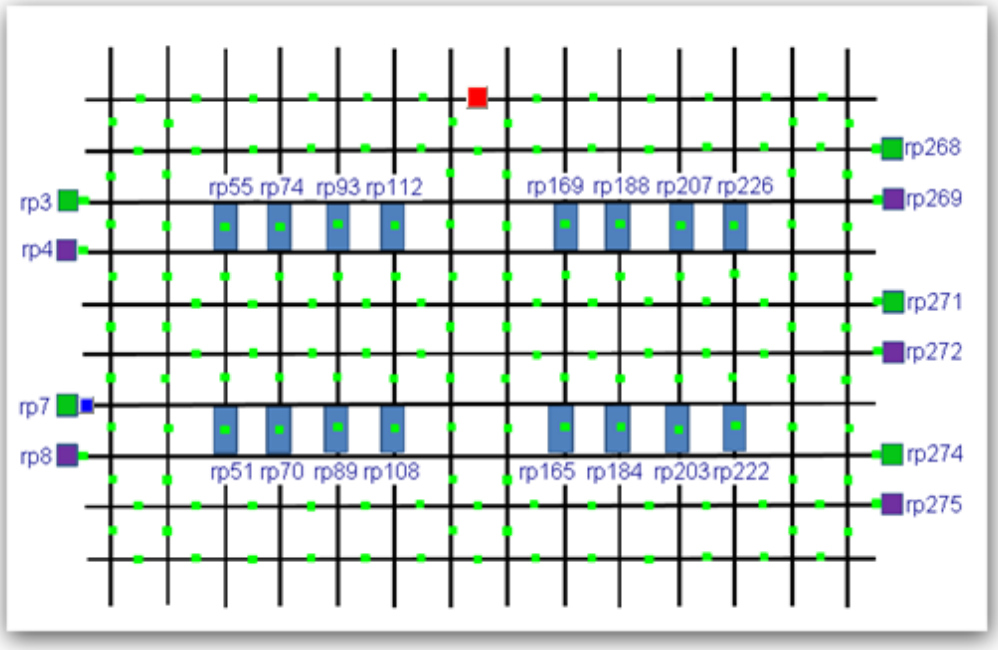

Figura 160. Ejemplo de utilización de la aplicación(10)

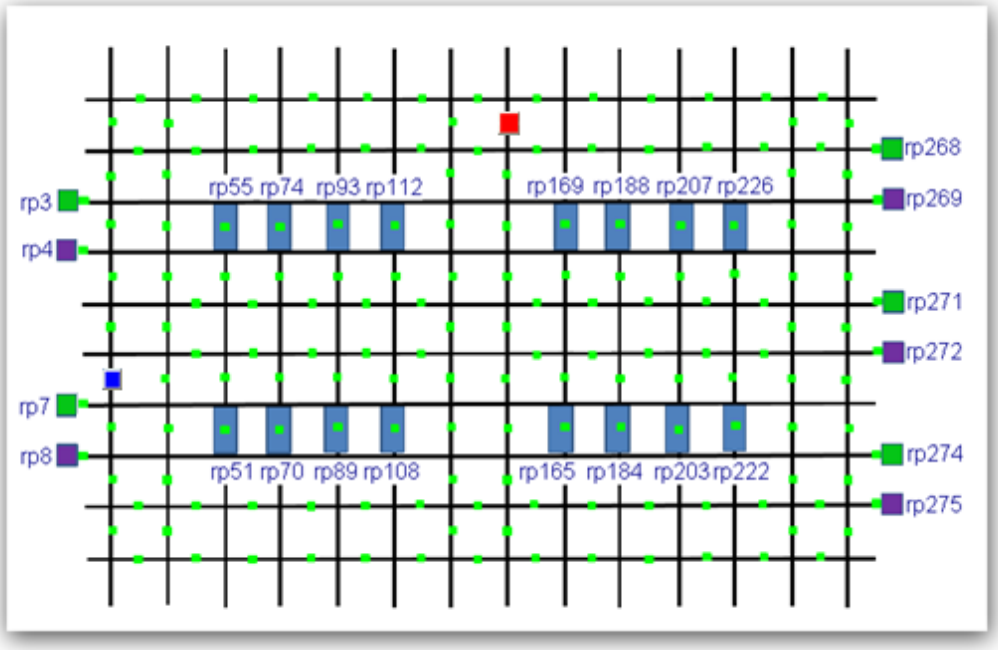

Figura 161. Ejemplo de utilización de la aplicación(11)

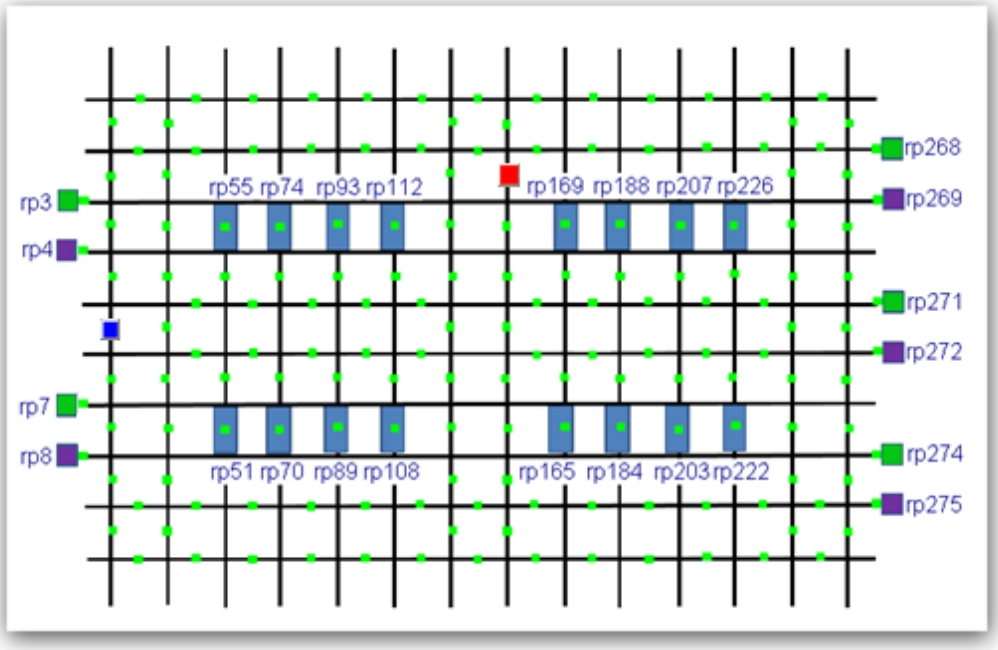

Figura 162. Ejemplo de utilización de la aplicación(12)

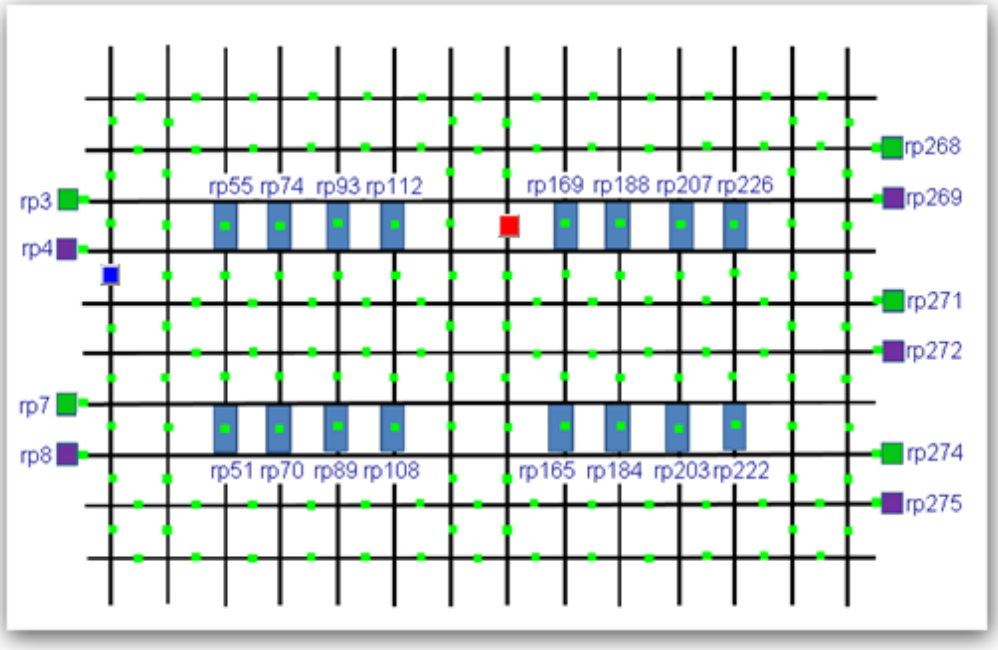

Figura 163. Ejemplo de utilización de la aplicación(13)

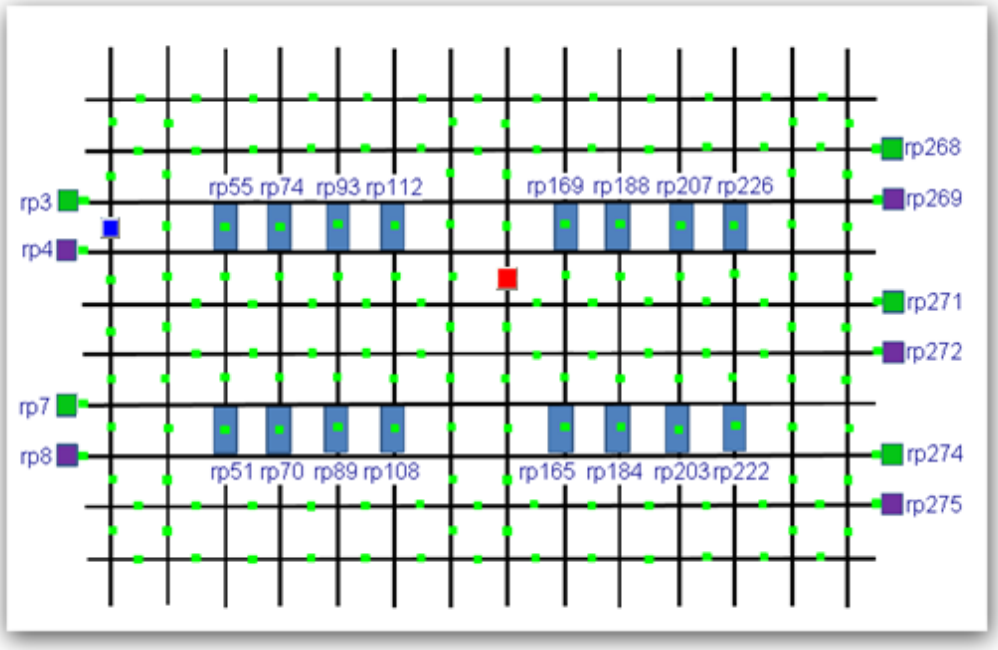

Figura 164. Ejemplo de utilización de la aplicación(14)

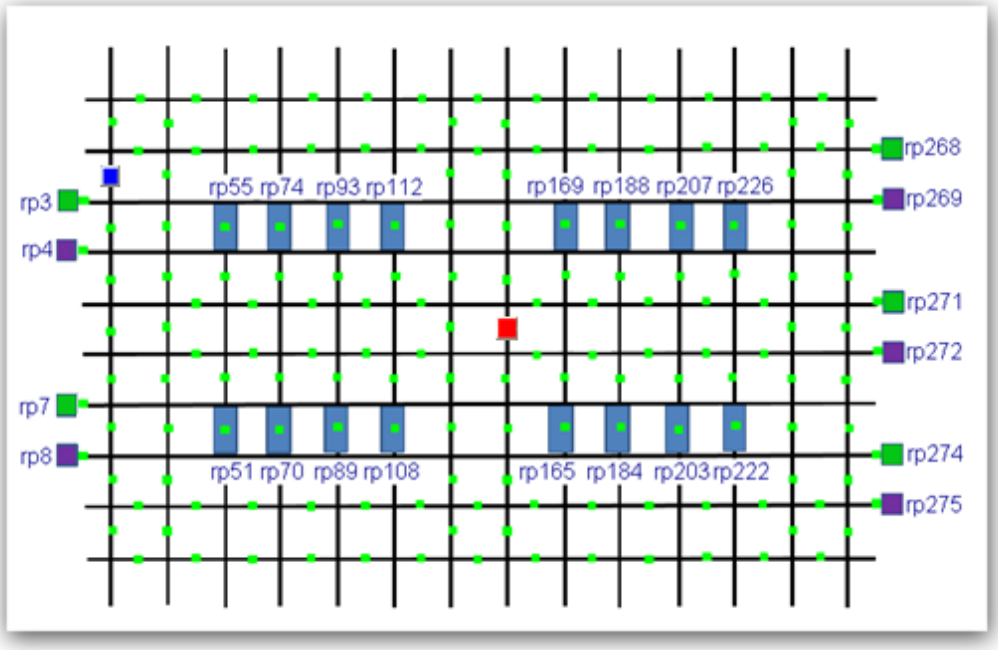

Figura 165. Ejemplo de utilización de la aplicación(15)

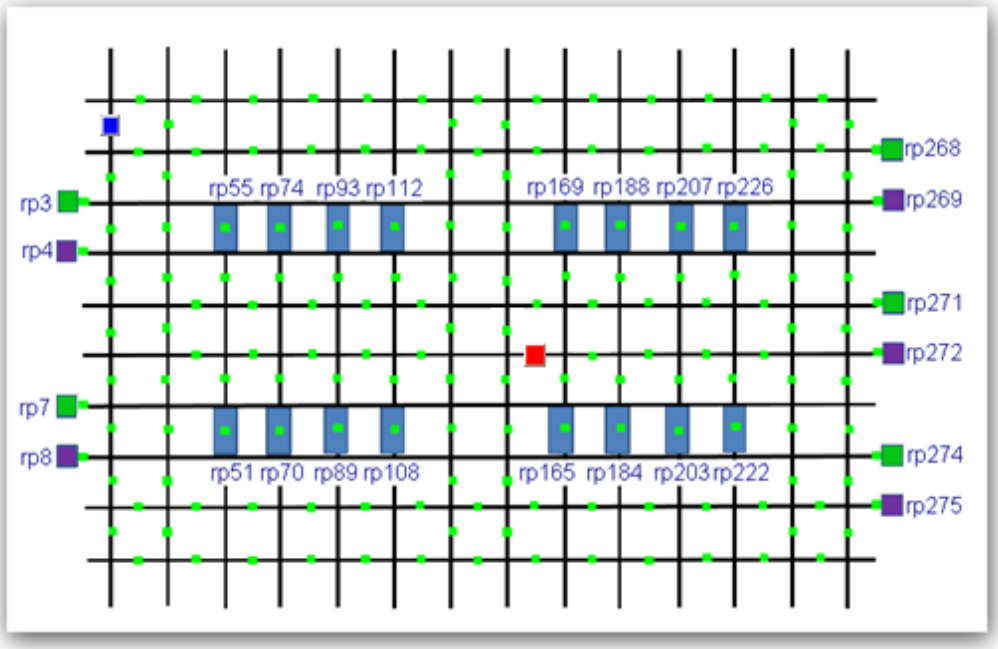

Figura 166. Ejemplo de utilización de la aplicación(16)

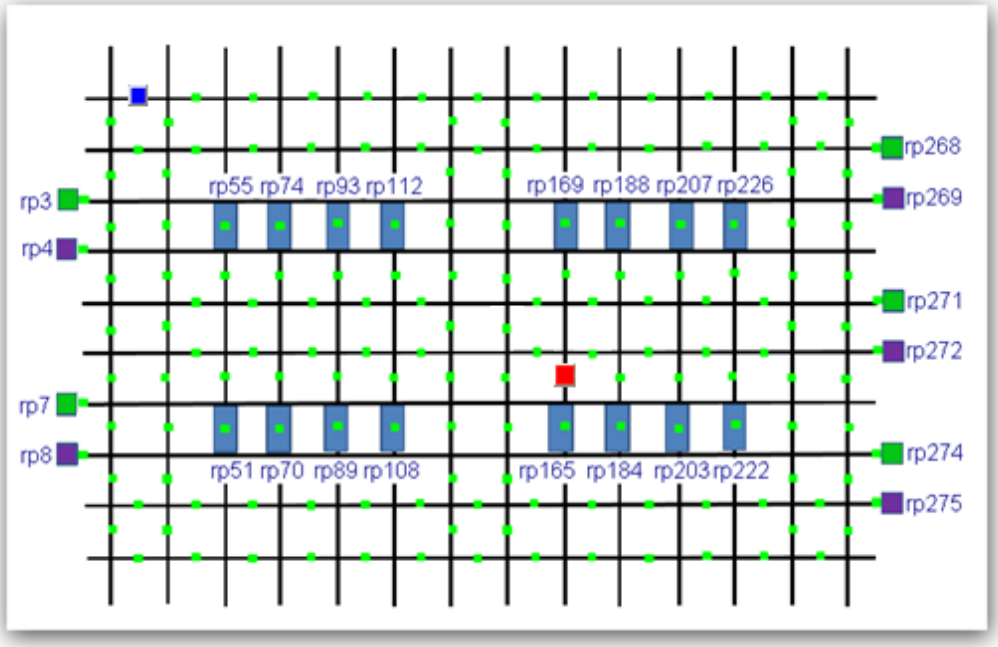

Figura 167. Ejemplo de utilización de la aplicación(17)

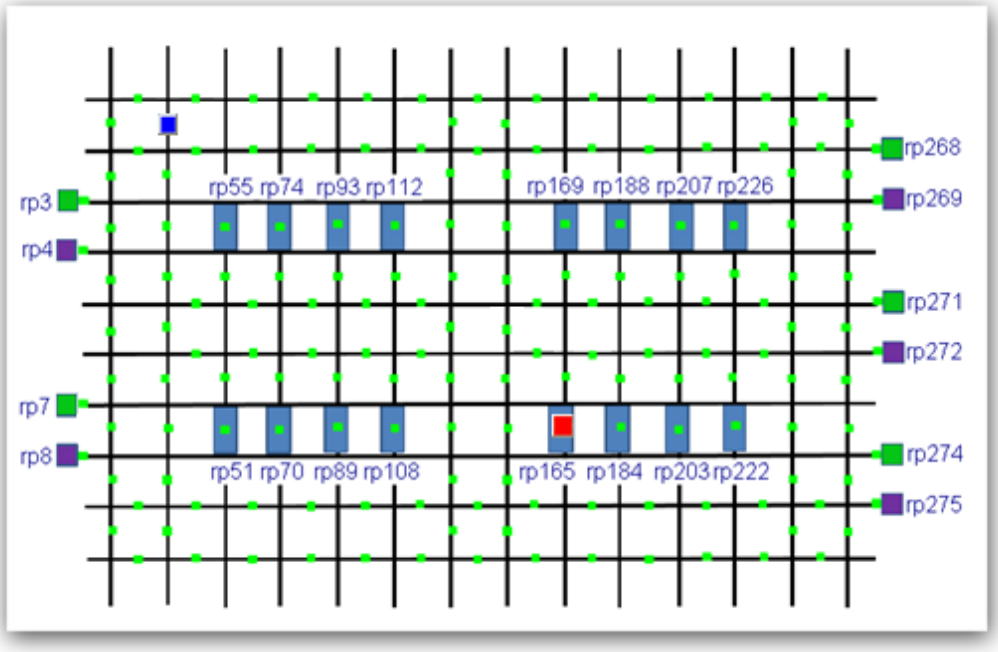

Figura 168. Ejemplo de utilización de la aplicación(18)

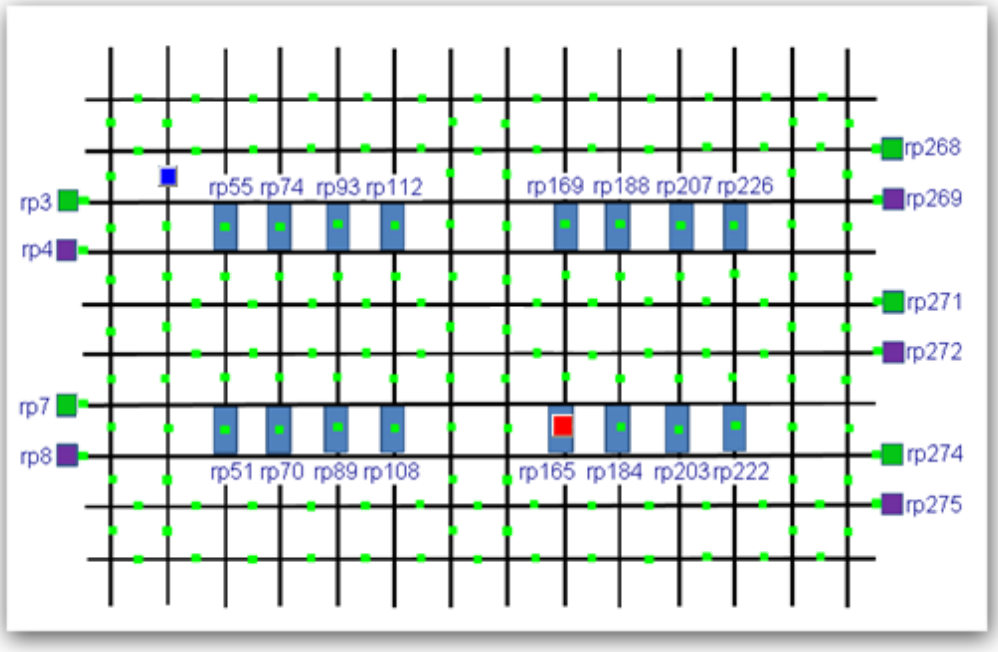

Figura 169. Ejemplo de utilización de la aplicación(19)

En este momento, el usuario decide realizar un nuevo pedido. Al encontrarse el robot 1 desocupado, la tarea se asigna directamente a él.

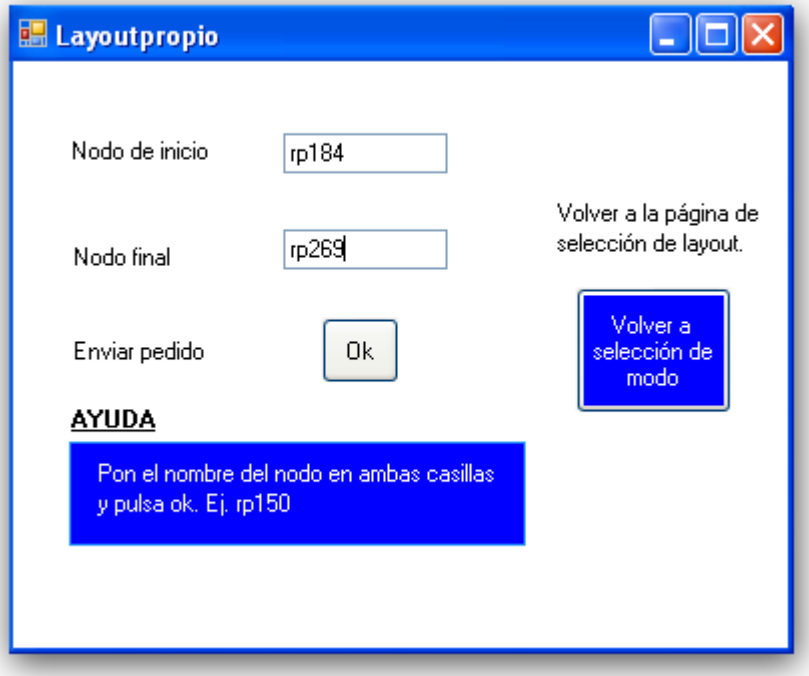

Figura 170. Envío de un tercer pedido

Desde ahora, el robot 1 debe ir a buscar un producto al almacén que se encuentra en el nodo "rp184" para llevarlo al tren de salida de producto "rp269".

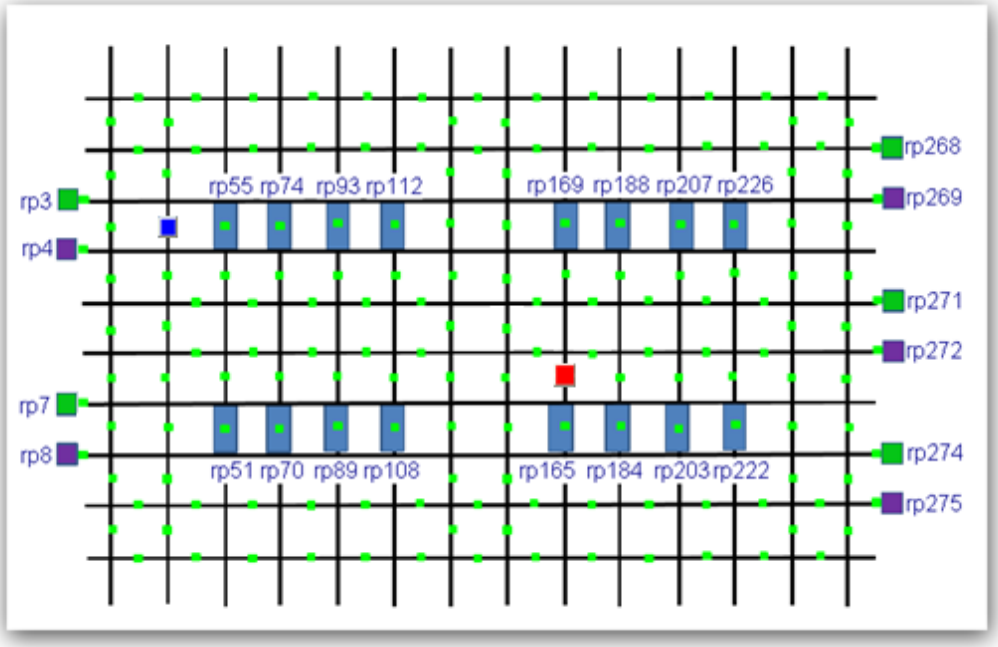

Figura 171. Ejemplo de utilización de la aplicación(20)

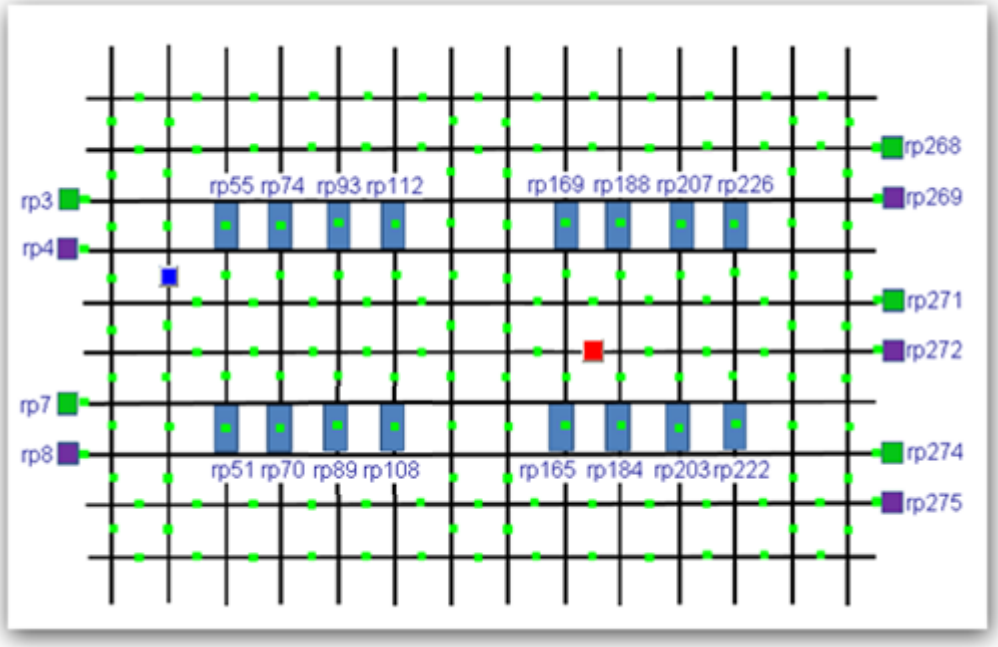

Figura 172. Ejemplo de utilización de la aplicación(21)

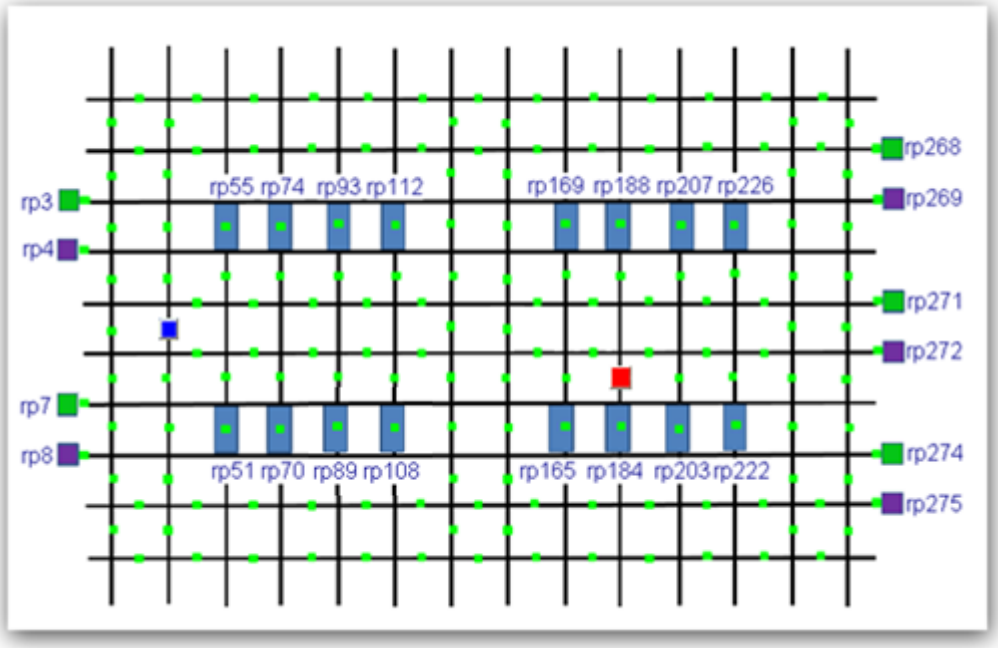

Figura 173. Ejemplo de utilización de la aplicación(22)

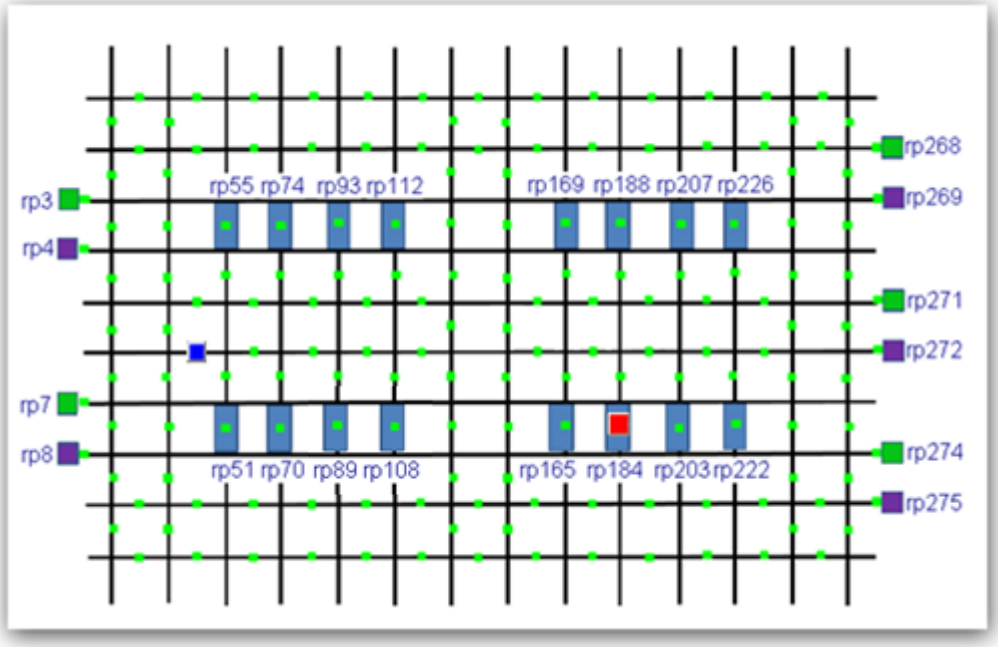

Figura 174. Ejemplo de utilización de la aplicación(23)

En este momento, mientras los robots 1 y 2 se encuentran ocupados, el usuario vuelve a enviar otro pedido, esta vez una entrada de producto por el tren de entrada "rp274" y se debe llevar el producto al almacén "rp226". Tal y como se muestra en la siguiente figura

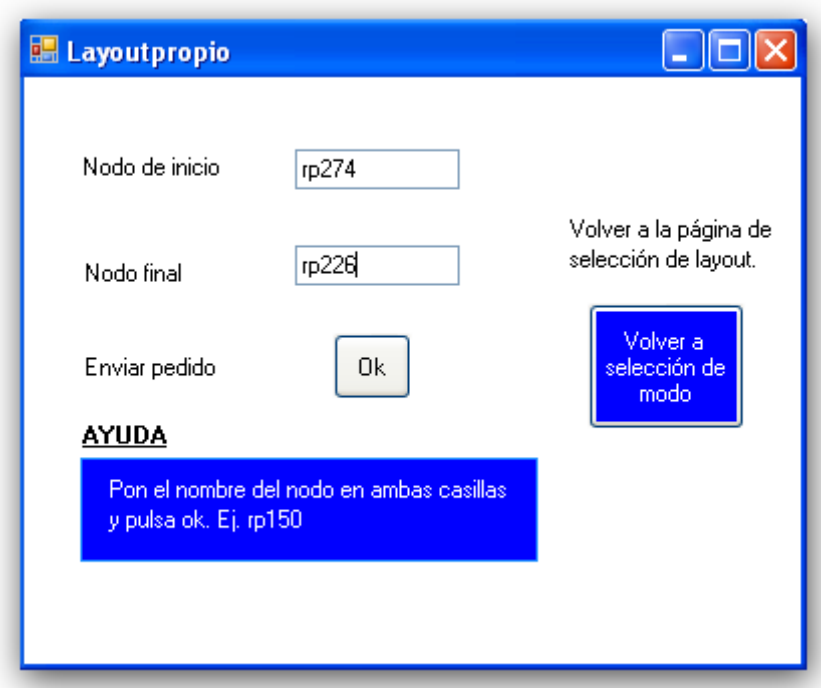

Figura 175. Introducción del cuarto y último pedido

A partir de este momento entra en acción el tercer robot.

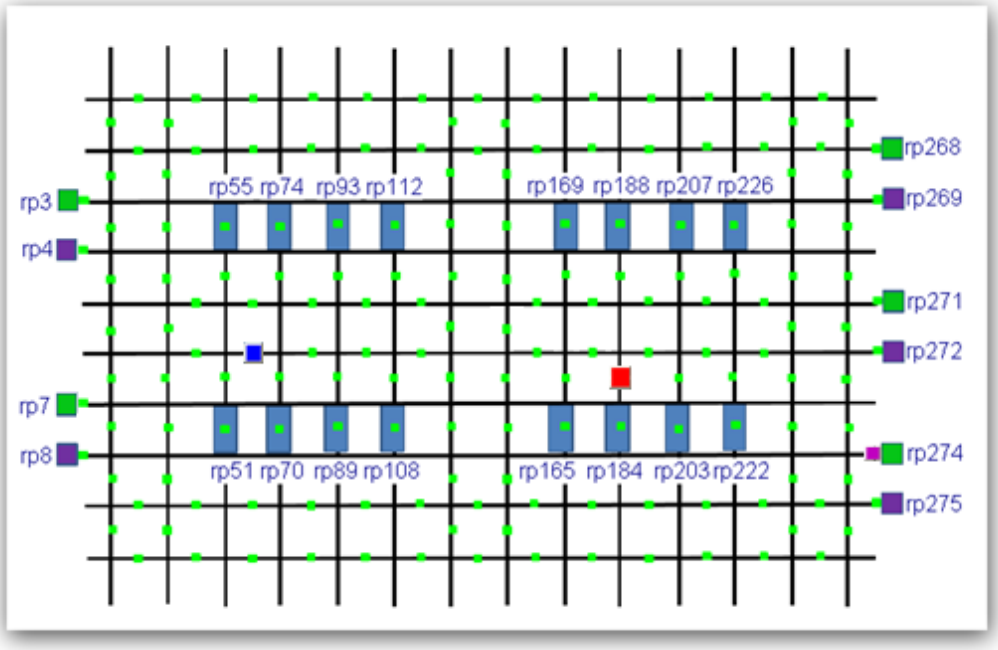

Figura 176. Ejemplo de utilización de la aplicación(24)

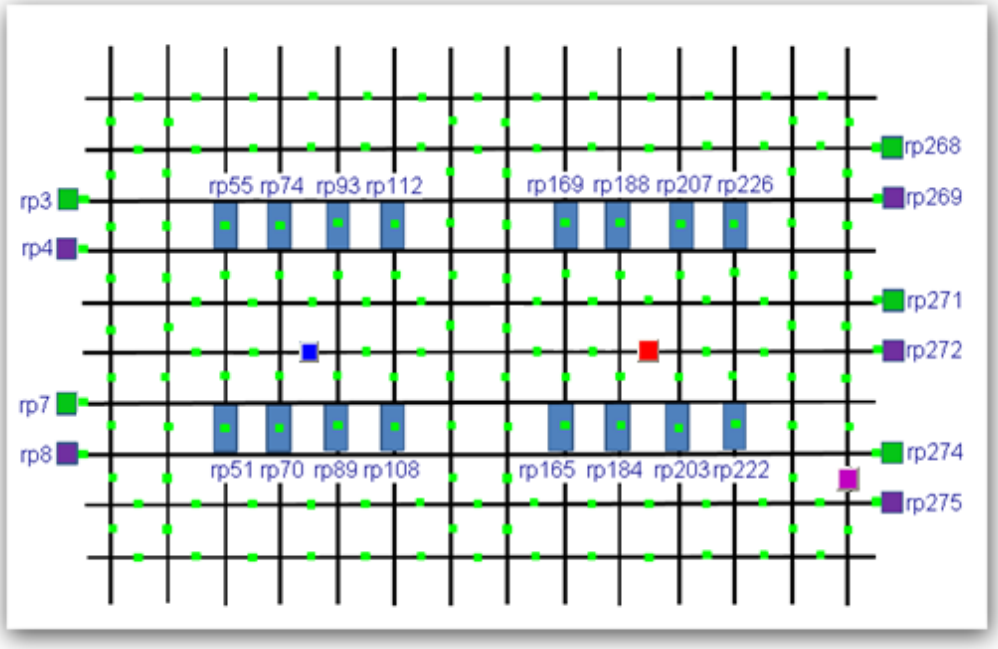

Figura 177. Ejemplo de utilización de la aplicación(25)

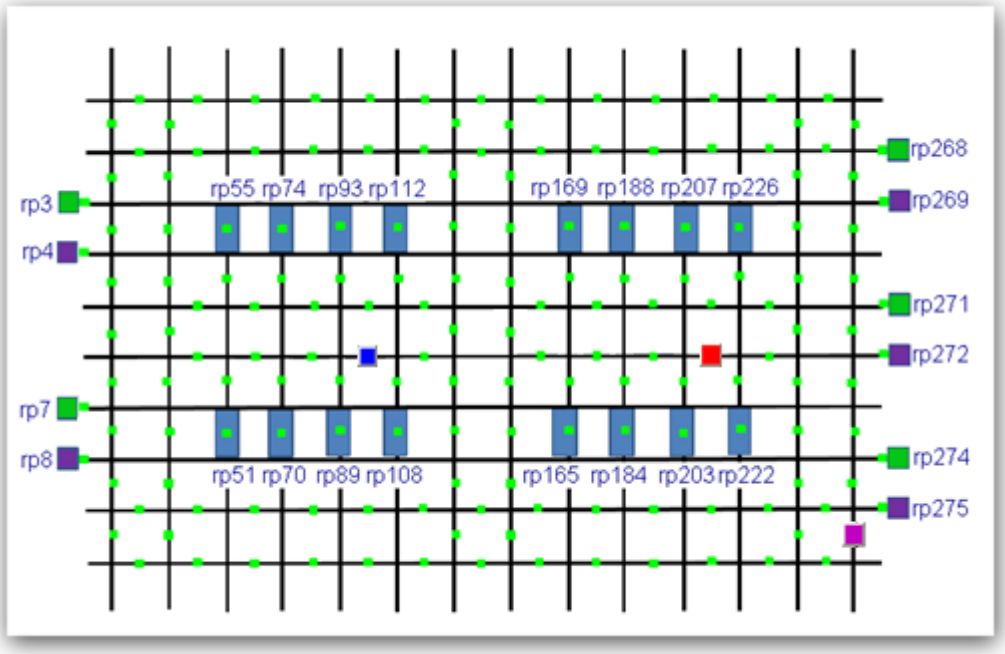

Figura 178. Ejemplo de utilización de la aplicación(26)

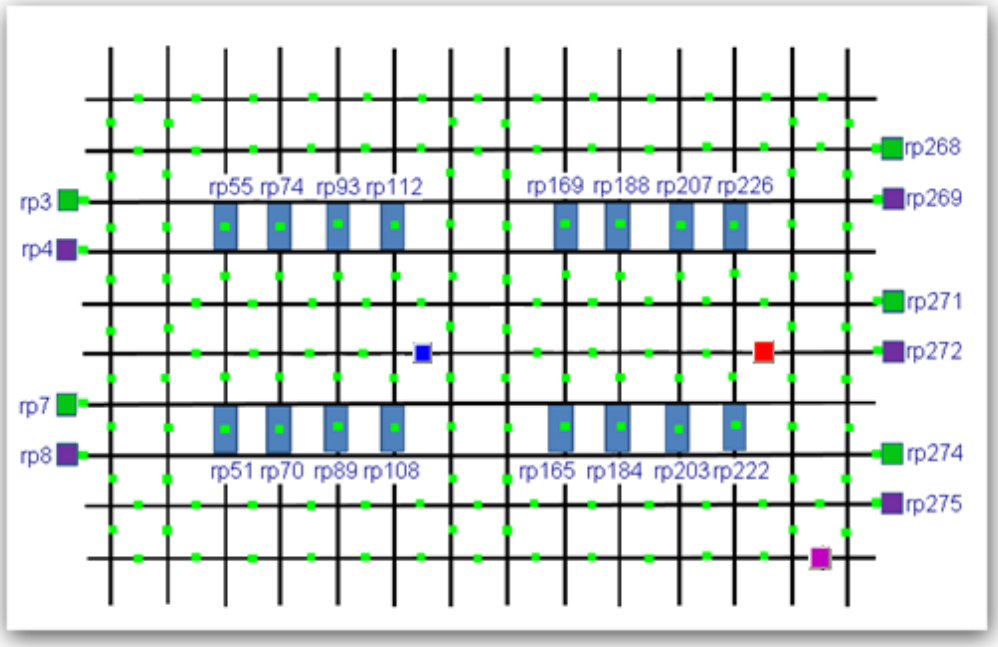

Figura 179. Ejemplo de utilización de la aplicación(27)

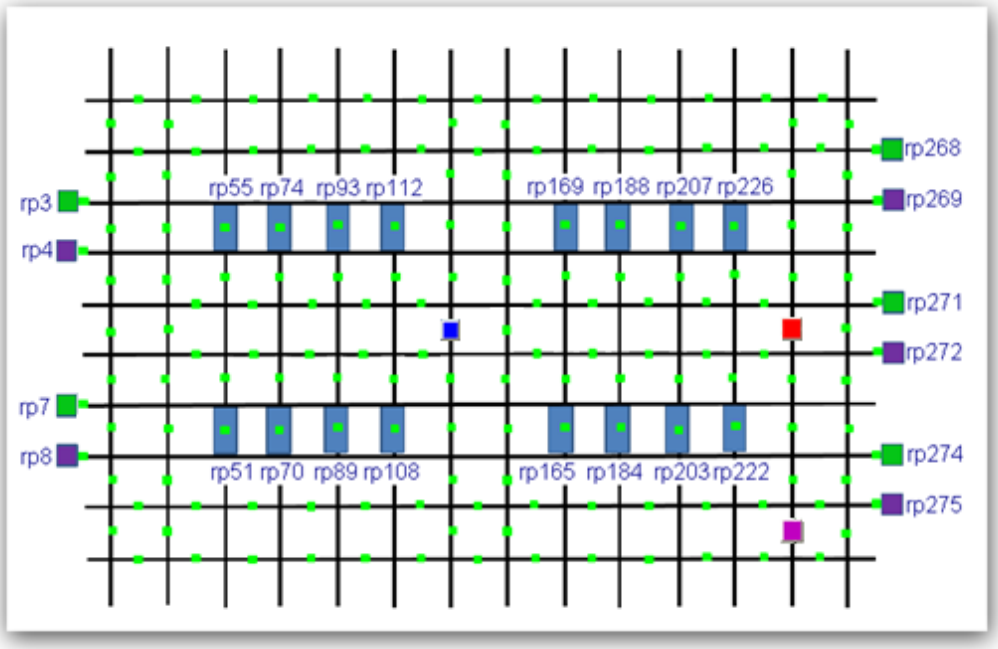

Figura 180. Ejemplo de utilización de la aplicación(28)

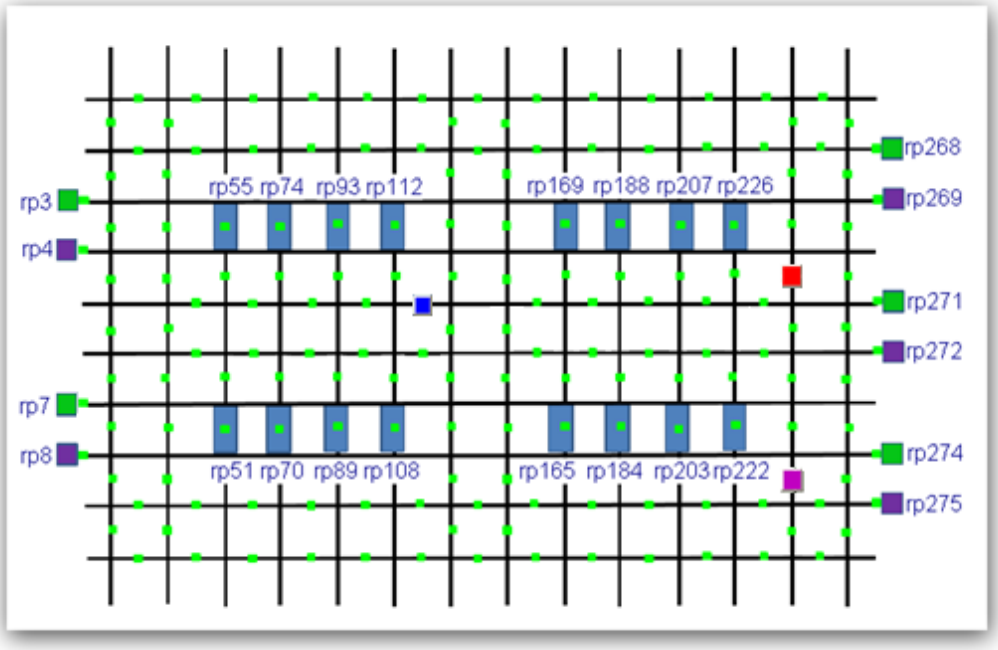

Figura 181. Ejemplo de utilización de la aplicación(29)

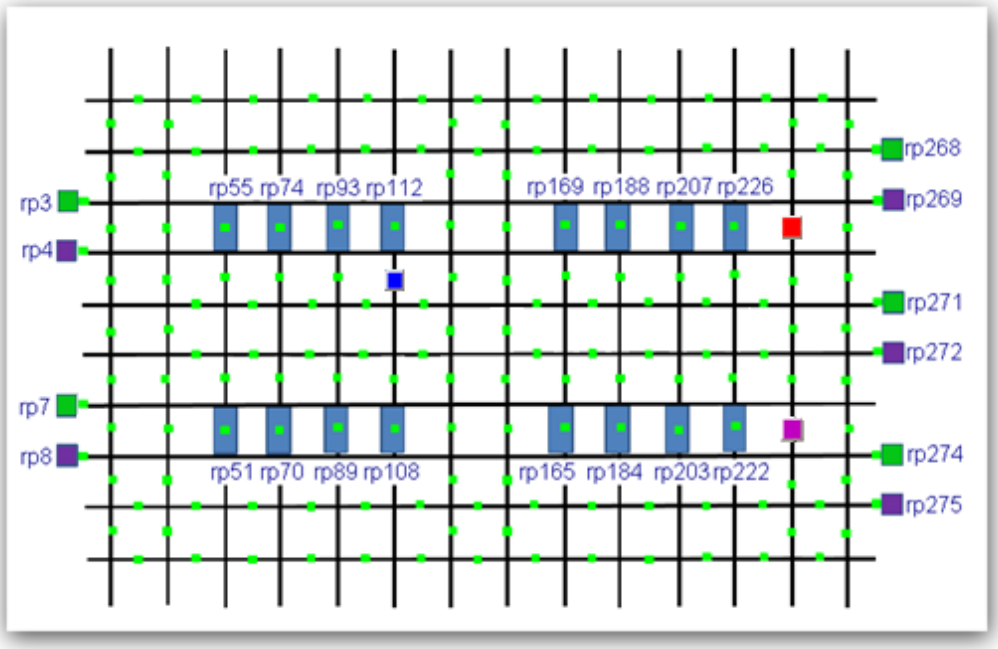

Figura 182. Ejemplo de utilización de la aplicación(30)

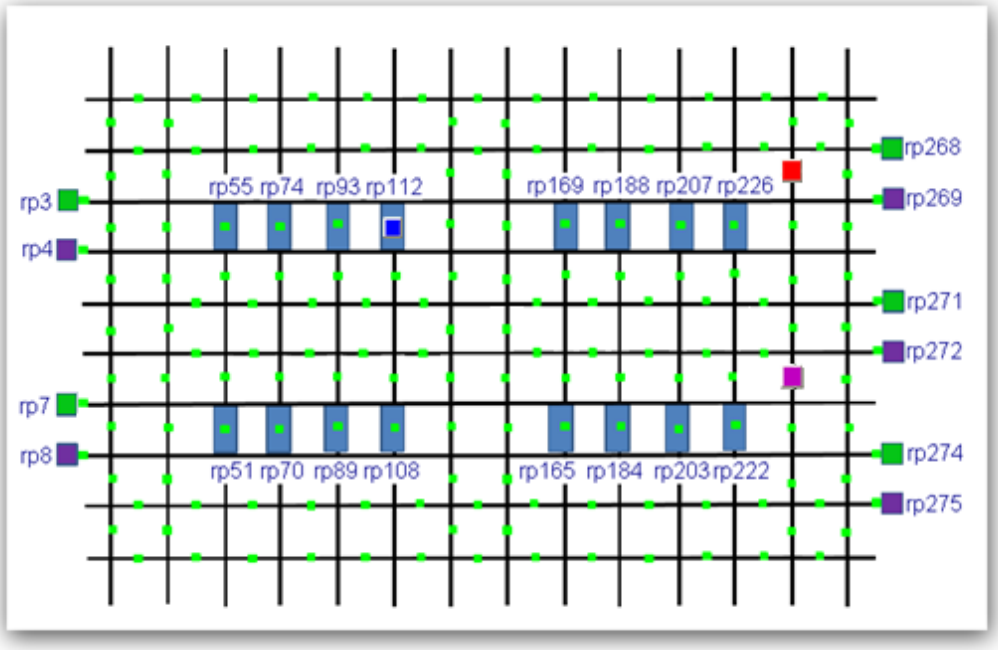

Figura 183. Ejemplo de utilización de la aplicación(31)

El robot 2 acaba de llegar a su destino, y por tanto ya no volverá a moverse hasta nueva orden.

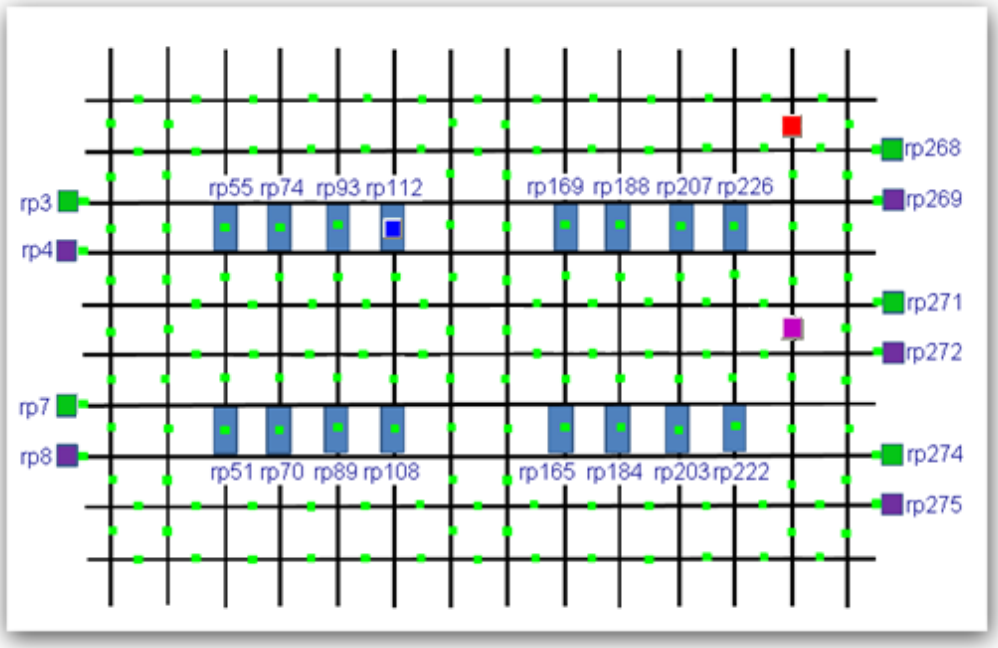

Figura 184. Ejemplo de utilización de la aplicación(32)

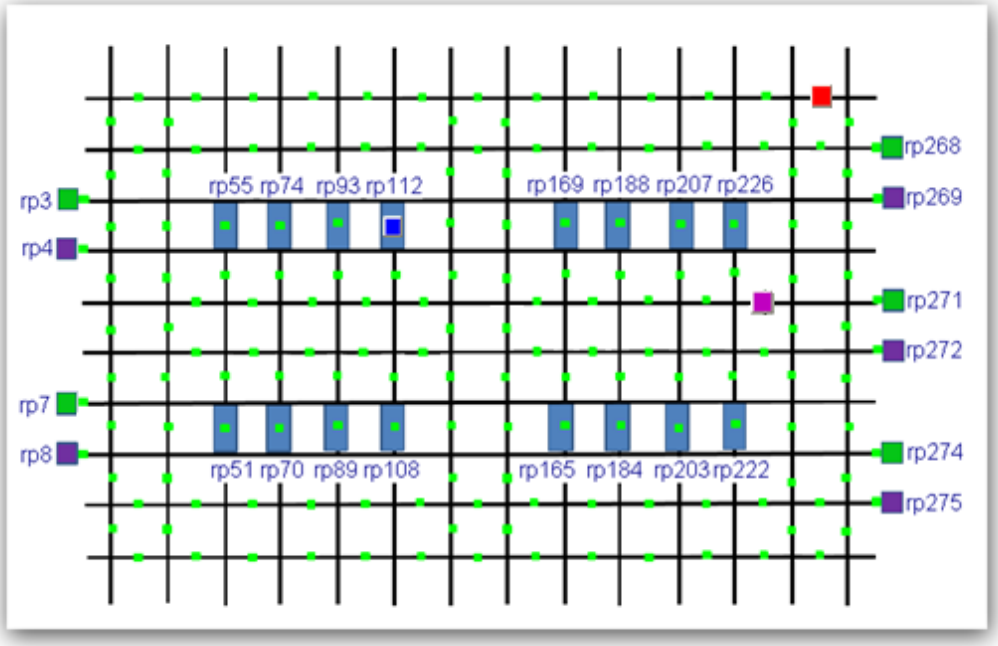

Figura 185. Ejemplo de utilización de la aplicación(33)

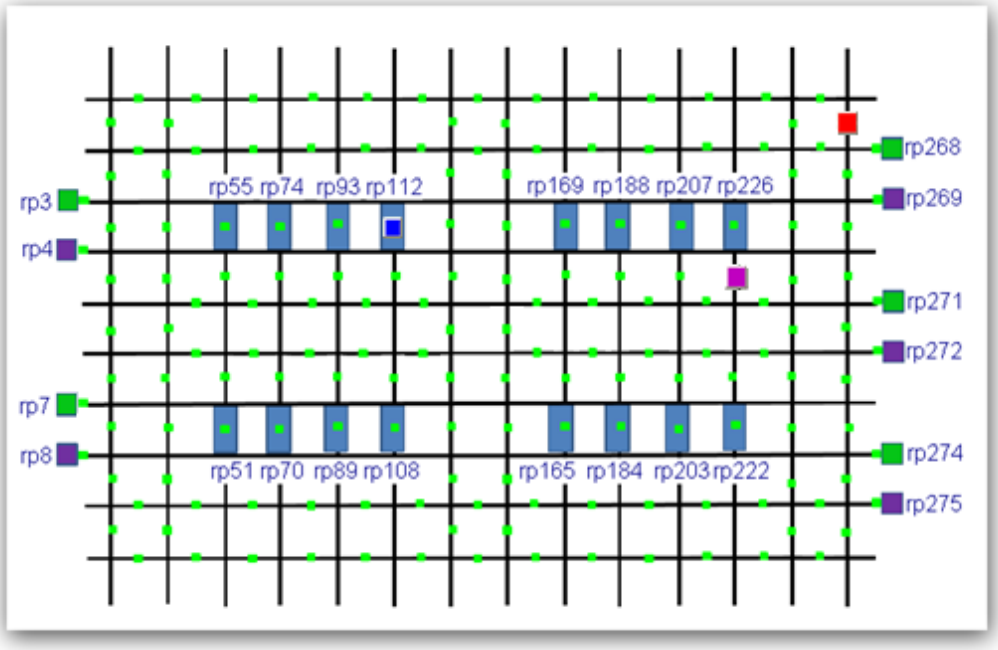

Figura 186. Ejemplo de utilización de la aplicación(34)

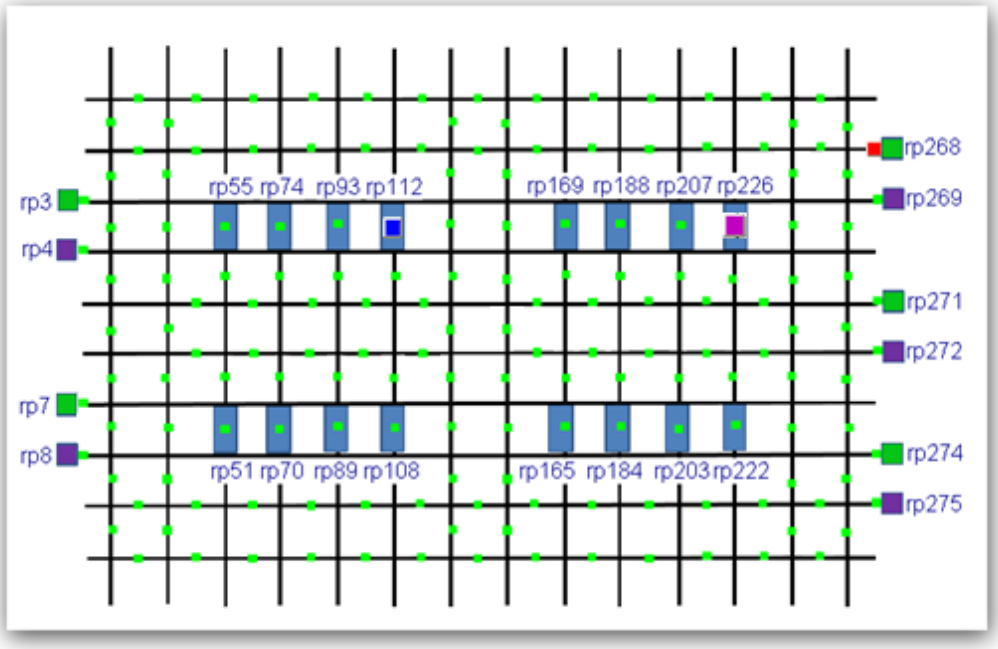

Figura 187. Ejemplo de utilización de la aplicación(35)

Finalmente, todos los robots han llegado a su destino y se mantienen a la espera de que el usuario vuelva a enviar una petición.

14. Resultado y conclusiones

En este apartado se describirán los aspectos más importantes a comentar en cuanto a resultados y posibles conclusiones a extraer.

En primer lugar, es necesario hacer mención al hecho de que, debido a la complejidad del proyecto, para ser capaz de llevarlo a cabo, se ha necesitado recibir lecciones de asignaturas específicas de master. Así como formación en todos y cada uno de los campos y aspectos que en este proyecto han sido tratados, puesto que un ingeniero industrial no recibe dicha formación durante la carrera.

En segundo lugar, haciendo mención a la estructura de software utilizada, debido a la falta de datos en cuanto a la creación de una aplicación de simulación de procesos automatizados, se ha tenido que adoptar una solución completamente nueva, debido a la inexistencia de aplicaciones similares. Gracias a las recomendaciones y enseñanzas del tutor Juan Carlos Hernández, se consiguió generar una estructura de datos única en el mercado. En la fase en la que se encuentra el proyecto, no es posible sacar partido a dicha solución, pero es posible que en futuras ampliaciones se consiga dar a la aplicación una funcionalidad mayor de cara a una posible comercialización.

Gracias al tiempo invertido en bases de datos SQL durante el periodo de aprendizaje, se consiguió optimizar mínimamente el código notando resultados satisfactorios. No obstante, con un mayor nivel de comprensión y de utilización del lenguaje SQL, sería posible mejorar la velocidad de iteración de la aplicación en gran medida.

Se puede concluir entonces, que se ha conseguido cumplir con las especificaciones básicas sin olvidar los límites del alcance de proyecto, aunque quizás el tiempo dedicado para conseguirlo haya sido mayor de lo esperado.

En cuanto a una conclusión personal, es muy grande la satisfacción que se recibe cuando se finaliza un proyecto que ha sido 100% confeccionado por uno mismo(sin olvidar, por supuesto, la importante aportación del tutor Juan Carlos Hernández). Después de prácticamente año y medio de trayectoria, se ha conseguido dar forma a algo que no existía, algo de lo que únicamente había una idea.

15. Propuestas de mejora

Como en cualquier otro proyecto, siempre se trabaja en contra del tiempo, por lo que se generan una serie de posibles mejoras que son visibles mientras se está desarrollando el proyecto, pero que es obvio desde el principio que no será posible implementarlas. Así pues, a continuación se detallan algunas posibles propuestas de mejoras futuras.

En primer lugar, debería optimizarse la aplicación de VB.net, reduciendo el código utilizado haciendo uso de los procesos almacenados de SQL. Se descubrió esta posibilidad en una fase muy avanzada del proyecto y no era viable reestructurar toda la programación del mismo. Este cambio haría que el rendimiento del programa aumentase considerablemente.

Por otro lado, otra propuesta de mejora sería realizar una interfaz gráfica mejorada. Esta interfaz gráfica vendría relacionada con el apartado de creación de un nuevo layout. El próximo paso a seguir, hubiese sido dar a la aplicación la opción de realizar una imagen de dicho layout para posteriormente trasladarla a la pantalla de SCADA. A su vez, dando opción a la actuación de un número variable de robots, en vez de uno fijo.

Posteriormente, se debería aplicar la posibilidad de realizar una entrada de información mediante un archivo .txt. Ahora mismo es el usuario el que debe realizar la acción de enviar uno por uno cada uno de los pedidos, mientras que de esta otra manera únicamente sería necesario introducir un documento .txt para hacer funcionar el sistema.

El objetivo de la creación de este proyecto es su aplicación en la industria, por lo que se debería aplicar el proyecto a cualquier nave industrial, mirando de complementarlo con otras aplicaciones como lectores, que simplemente leyendo el código de un producto determine donde debe ir ese producto (esto vendría a corresponderse con el archivo .txt comentado con anterioridad para la simulación).

Referencias bibliográficas

[1] RÍOS SEPULVEDA, L.A., Ingeniería: Testimonio de la historia humana, publicado el 7 de abril de 2013. [En línea]

[http://introingenieriausach.blogspot.com.es/2013/04/cohete-v2-el-cohete-v2-especificamente.html, 31 de mayo de 2014]

[2] GASPERI, Michael, Machina Speculatrix. [En línea]

[http://www.extremenxt.com/walter.htm, 31 de mayo de 2014]

[3] CATPHI, Robots, publicado el 25 de setiembre de 2010. [En línea]

[http://catphi.wordpress.com/2010/09/25/robots/, 31 de mayo de 2014]

[4] ALEX, The Hopkins Beast, publicado el 22 de diciembre de 200 8. [En línea]

[http://www.weirduniverse.net/blog/comments/the_hopkins_beast/, 31 de mayo de 2014]

[5] RB ROBOTICS, RB5X Robot "The intelligent robot", actualizado el 1 de octubre de 2008. [En línea]

[http://www.theoldrobots.com/rb5x.html, 31 de mayo de 2014]

[6] FERNANDEZ, M.A., FERNANDEZ, D., VALMASEDA, C., Planificación de trayectorias para un robot móvil. [En línea]

[eprints.ucm.es/11301/1/MemoriaProyectoSSII.pdf, 31 de mayo de 2014]

[7] WIKIPEDIA, Sojourner, actualizado el 11 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/Sojourner_(rover), 31 de mayo de 2014]

[8] WIKIPEDIA, Mars exploration rover, actualizado el 3 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/Mars_Exploration_Rover, 31 de mayo de 2014]

[9] ULANOF, Lance, Sony AIBO ERS-7M2, publicado el 15 de diciembre de 2004. [En línea]

[http://www.pcmag.com/article2/0,2817,1742094,00.asp, 31 de mayo de 2014]

[10] WIKIPEDIA, Robotic lawn mower, actualizado el 10 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/Automower#Husqvarna_Automower, 31 de mayo de 2014]

[11] PARKER, Wendy, Preventing Roomba suicide, publicado el 21 de junio de 2013. [En línea]

http://www.overdriveonline.com/preventing-roomba-suicide/, 31 de mayo de 2014]

[12] WIKIPEDIA, Mobile Robot, actualizado el 24 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/Mobile_robot#History, 31 de mayo de 2014]

[13] Dr. ANIBAL OLLERO, Planificación de trayectorias para robots móviles, tesis doctoral presentada el 5 de julio de 1995. [En línea]

[http://webpersonal.uma.es/~VFMM/tesis.html, 31 de mayo de 2014]

[14] WIKIPEDIA, Algorithm, actualizado el 31 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/Algorithm, 31 de mayo de 2014]

[15] ARCOS SANCHEZ, J.A. , Sistema de navegación y modelado del entorno para un robot móvil, Noviembre de 2009. [En línea]

[tierra.aslab.upm.es/documents/PFC/PFC_JAArcos.pdf, 31 de mayo de 2014]

[16] MARTÍNEZ LUGO, J.A., La pila OSI, publicado el 6 de marzo de 2009. [En línea]

[http://martinezlugoalfonso.blogspot.com.es/2009/03/modelo-osi.html, 31 de mayo de 2014]

[17] El modelo OSI y los protocolos de red, capítulo 2. [En línea]

[es.scribd.com/doc/113859431/pila-OSI, 31 de mayo de 2014]

[18] JIMENEZ, J.A., OVALLE, D.A., BRANCH, J.W., "Comunicación en sistemas de múltiples robots desde la metodología MAD-Smart" . Ingeniería e investigación, vol. 28, nº 2 (2008), p. 59-65 [En línea]

[http://www.redalyc.org/pdf/643/64328209.pdf, 31 de mayo de 2014]

[19] MATRIKONOPC, OPC Video Tutorials. [En línea]

http://www.matrikonopc.com/training/opc-multimedia-tutorial-new/index.html, 31 de mayo de 204]

[20] KOMINEK, D., OPC: The Ins and Outs to What It's About, Canada 2009. [En línea]

[https://www.matrikonopc.com/downloads/434/index.aspx, 31 de mayo de 2014]

[21] NATIONAL INSTRUMENTS, Introduction to OPC, publicado el 7 de setiembre de 2012. [En línea]

[http://www.ni.com/white-paper/7451/en/, 31 de mayo de 2014]

[22] WIKIPEDIA, OPC, actualizado el 28 de mayo de 2014. [En línea]

[http://es.wikipedia.org/wiki/OPC, 31 de mayo de 2014]

[23] MELLADO ARTECHE, M., Modelado, Detección de Colisiones y planificación de movimientos en sistemas robotizados mediante volúmenes esféricos, Valencia (1996). [En línea]

[robotica.isa.upv.es/pub/tesis_mmellado.pdf, 31 de mayo de 2014]

[24] E.S.H. HOU, D. ZHENG, "Mobile Robot Path Planning Based On Hierarchical Hexagonal Decomposition And Artificial Potential Fields". Journal Of Robotics Systems, 11(7), 1994.

[25] YANDÚN, A., SOTOMAYOR, N., "Planeación y seguimiento de trayectorias para un robot móvil", Ecuador.

[http://bibdigital.epn.edu.ec/bitstream/15000/4913/1/Planeaci%C3%B3n%20y%20seguimiento%20de%20trayectorias.pdf, 31 de mayo de 2014]

[26] WIKIPEDIA, Floyd-Warshall algorithm, actualizado el 19 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm, 31 de mayo de 2014]

[27] DAPENA, E., Modelado del entorno en robótica móvil. Presentación de clase, Universidad de los Andes.

[28] VIVAS, C., Detección de colisiones y planificación de caminos, Apuntes de clase Automatización y Robótica 2 (2006). [En línea]

[http://www.esi2.us.es/~vivas/ayr2iaei/DET_PLAN_CAMINOS.pdf, 31 de mayo de 2014]

[29] WIKIPEDIA, Scada, actualizado el 20 de mayo de 2014. [En línea]

[http://en.wikipedia.org/wiki/SCADA#Supervisory_station, 31 de mayo de 2014]

[30] LORENZO LLEDÓ, G., Automatización de una planta industrial, tesis doctoral (2007). [En línea]

[http://rua.ua.es/dspace/bitstream/10045/10056/1/Suficiencia%20Gonzalo.pdf, 31 de mayo de 2014]

[31] CHACON,D., DIJORT, O., CASTRILLO, J., Supervisión y control de procesos, EUPVG-UPC (2001-2002). [En línea]

[http://ocw.upc.edu/sites/default/files/materials/15012628/40194-3452.pdf, 31 de mayo de 2014]

[32] SANTANDER, M.H., Authentication Issues between entities during protocol message exchange in SCADA Systems, publicado el 20 de agosto de 2012. [En línea]

[http://www.tuicool.com/articles/YRveyy, 31 de mayo de 2014]

[33] PEREIRA, G., iFix Fundamentals, GE Fanuc. [En línea].

[http://www.scribd.com/doc/120563222/iFix, 31 de mayo de 2014]

[34] OPCTECHS, Configuration Manual for SCADA iFix v4.0. [En línea]

[http://www.opctechs.com/edoc/SCADA/IFix.pdf, 31 de mayo de 2014]

[35] KEPWARE TECNOLOGIES, KEPServerEX Client Connectivity Guide for GE's Proficy iFix, October 2011. [En línea]

[http://www.kepware.com/Support_Center/SupportDocuments/KTSM00017_IFix_Connectivity_Guide.pdf, 31 de mayo de 2014]

[36] HERNANDEZ PALACÍN, J.C., Apuntes de clase de Fabricación Automatizada y Robótica Industrial, curso 2012-2013.